Application Specification for Microsoft® Windows® 2000

SERVER

ADVANCED SERVER

DATACENTER SERVER

Design Guide for Building Server Applications

VERSION 1.2
December 8, 1999

Microsoft Corporation

Acknowledgments

Microsoft wishes to thank the following organizations for their technical review, feedback, and consideration in the drafting of this specification. Their feedback was invaluable to help guide the content of this specification to better meet the needs of enterprise customers.

American International Group, Inc.

The Baan Company

BMC Software

Carnegie Mellon University

Charles Schwab & Co., Inc.

CIGNA

Credit Suisse First Boston

Ford Motor Company

Massachusetts Institute of Technology

Nortel Networks

Pfizer, Inc.

The Taylor Group

The University of Texas at Austin – The Graduate School of Business

Warburg Dillon Read

Welcome

The Application Specification for Windows 2000 was developed by Microsoft in cooperation with customers and third-party developers to identify the technical definition of the model business application for Windows 2000. This specification will help software developers take advantage of the new technologies in Windows 2000 so that your application is more manageable, more reliable, and reduces cost of ownership for your customers. It is intended for developers of all kinds, including independent software vendors and corporate application developers.

The specification has two versions. This document details the requirements for multi-user server applications. There is a similar specification for building single-user (primarily desktop) applications, available at http://msdn.microsoft.com/certification.

Top reasons your customers will benefit

An application that meets this specification…

- Supports a globally available infrastructure for easier access for the user and easier management for the administrator.
- Provides secure access to resources and security for client server interactions.
- Is ready to run on a server cluster (for applications that certify on Windows 2000 Advanced Server and Datacenter Server). By exploiting Cluster service features an application can minimize the service downtime caused by system failures or planned server maintenance and upgrades.

Certified for Windows

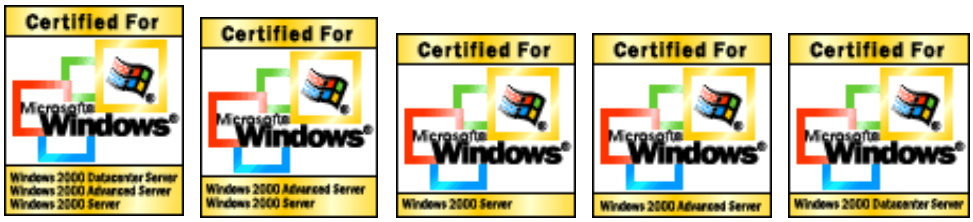
The Application Specification for Windows 2000 defines the technical requirements for applications to earn the “Certified for Microsoft Windows” logo. Applications may carry the “Certified for Microsoft Windows” logo, once they have passed compliance testing and have executed a logo license agreement with Microsoft. This logo lets your customers know that your application offers a high quality computing experience available on Windows.

Windows Certification for server applications is available for any of the following operating systems:

- Windows 2000 Server
- Windows 2000 Advanced Server
- Windows 2000 Datacenter Server*

* Note: Windows 2000 Datacenter Server is on a different release schedule than the other Windows 2000 operating systems. Microsoft is currently evaluating whether any additional requirements specific to Datacenter will be required for Certification on Windows 2000 Datacenter server.

The logo you receive will indicate the version(s) of Windows for which your product is certified. Samples logos are shown below.



Compliance testing for the Windows Certification program is performed by VeriTest, an independent testing lab. Compliance testing will be done using the latest released versions of Windows 2000, including any service packs.

For additional information on how to test for the “Certified for Windows” logo, see http://msdn.microsoft.com/certification.

Note: To simplify the logo programs for both customers and ISVs, the requirements for the Windows 2000 Server logo will over time equal the core baseline BackOffice logo requirements. For more information on the current BackOffice logo, see http://www.microsoft.com/backoffice/designed.

References

Resource	*Address*
Desktop Application Specification for Windows 2000	http://msdn.microsoft.com/certification/
“Certified for Windows” Logo Program	http://msdn.microsoft.com/certification/
VeriTest Logo Lab email	LogoLab@veritest.com
Certified for Windows Logo email	Winlogo@microsoft.com
Windows 2000 Developer information	http://msdn.microsoft.com/windows2000
BackOffice Logo Program	http://www.microsoft.com/backoffice/designed
Knowledge Base	http://www.microsoft.com/support/
Microsoft Platform SDK (Software Developer Kit) Documents the Win32® and Win64™ application programming interfaces (APIs).	Provided with the Microsoft Developer Network (MSDN) Professional Subscription. To subscribe, visit http://msdn.microsoft.com/subscriptions/

Checklist for Windows Certification

Chapter 1 Windows Fundamentals

1.1 Perform primary functionality and maintain stability
1.2 Provide 32-bit components and document any 16-bit code
1.3 Support Long File Names and UNC paths
1.4 Support printers with long names and UNC paths
1.5 Do not read from or write to Win.ini, System.ini, Autoexec.bat or Config.sys
1.6 Ensure non-hidden files outside of your application directory have associated file-types, and all file-types have associated icons, descriptions and actions.
1.7 Perform Windows version checking correctly
1.8 Kernel mode drivers must pass verification testing
1.9 Hardware drivers must pass WHQL testing
1.10 Client components must comply with the Desktop Application Specification for Windows 2000

Chapter 2 Install//Uninstall *(Note: If your application uses the Windows Installer service, requirements 2.2 through 2.8 in this Server Specification are superceded by Chapter 2 of the Desktop Application Specification. Note: Desktop Requirement 2.6 is not required.)*

2.1 Do not attempt to replace files that are protected by Windows File Protection
2.2 Do not overwrite non-proprietary files with older versions
2.3 Install to Program Files by default
2.4 Install shared files to the correct locations
2.5 Refcount all shared application files during installation
2.6 Support Add/Remove Programs properly
2.7 Decrement the count on shared application files during uninstall
2.8 Ensure correct uninstall

Chapter 3 User Interface Fundamentals *(3.1 through 3.4 required only if any graphical UI is presented.)*

3.1 Support standard system size, color, font, & input settings
3.2 Ensure compatibility with the High Contrast option
3.3 Provide documented keyboard access to all features
3.4 Expose the location of the keyboard focus
3.5 Do not place shortcuts to documents, help or uninstall in the Start Menu

Chapter 4 Active Directory

4.1 Use Active Directory appropriately
4.2 Document the storage and replication impact of your application
4.3 Document your application's use of objects and attributes in the base schema
4.4 Information stored in Active Directory must be "globally interesting"
4.5 Attribute values must not exceed 500KB in length and the total object size must not exceed 1MB
4.6 If you extend the schema, adhere to Active Directory Schema extensibility rules

Chapter 5 Security Services

5.1 Document services that require more than User level privileges to run
5.2 Win32 clients running in the context of a trusted domain account must support Single Sign-On

Chapter 6 Cluster Service *(Required for Advanced Server and Datacenter* Certification)*

6.1 Applications must be able to install on 2 nodes for certification on Windows 2000 Advanced Server and 2,3, and 4 nodes on Windows 2000 Datacenter Server.
6.2 Applications must support failover to all cluster nodes
6.3 Clients must survive failure of server application without crashing or affecting the stability of the system

** Note: Windows 2000 Datacenter Server is on a different release schedule than the other Windows 2000 operating systems. Microsoft is currently evaluating whether any additional requirements specific to Datacenter will be required for Certification on Windows 2000 Datacenter server.*

Who is this Specification For?

This specification is intended for vendors and corporate developers building multi-user applications that run on any of the Windows 2000 Server family of operating systems. These distributed applications may be client/server applications or n-tier applications. This specification is specifically targeted at the server component of a distributed application. If you are building a single-tier application that is not a multi-user application, we suggest you consider designing to the Desktop Application Specification for Windows 2000.[1]

This specification is primarily focused on the middle-tier component(s), e.g. the component that (in most cases) runs on the server. Most, though not all, of the requirements in this specification focus on what the server component of an application should do and how it should behave. In a few cases this specification has client requirements that describe how client components that your application may have should behave when communicating with your server component. However, this specification does not attempt to describe complete requirements for applications (or components of applications) that are primarily intended to run on Windows 2000 Professional. Those requirements are described in the Desktop Application Specification for Windows 2000.

To qualify for the Certified for Windows logo, your application must comply with this Server Specification, and additionally, if your distributed application includes client components, then those client components must comply with the Desktop Application Specification for Windows 2000. Distributed applications that have client components are eligible to carry both Server and Professional designations on their Certified for Windows logo. Note that certain types of clients may be completely or partially exempted from complying with the Desktop Application Specification. See Requirement 1.10 and Appendix B for details.

Finally, if your application uses third party server components, see Appendix C.

[1] Available at http://msdn.microsoft.com/certification/appspec.asp

Contents

Acknowledgments iii

Welcome iv

Checklist for Windows Certification vi

Who is this Specification For? vii

Chapter 1 Windows Fundamentals 1
Describes the requirements for consistent, stable functionality.

Chapter 2 Install/Uninstall 12
Describes the requirements to help ensure that applications perform a clean installation and can be properly uninstalled.

Chapter 3 User Interface Fundamentals 18
Describes the requirements to provide an accessible user interface.

Chapter 4 Active Directory 27
Describes the requirements for integrating with Active Directory.

Chapter 5 Security Services 36
Describes the requirement to document the security context that your application runs in and the requirement to support Single Sign-On.

Chapter 6 Cluster Service 43
Describes requirements for your application to run in clustered environment to improve availability of your application.

Appendix A Best Practices 50
Best practices are not required for Certification on Windows 2000, but are strongly encouraged to provide a better user experience.

A-1 Do not require a reboot for installation of your application on Windows 2000
A-2 Use SHGetFolderPath to determine special folder paths
A-3 Test your application under Terminal Services
A-4 Globalization
A-5 Localizability
A-6 Use 64-bit compatible data types
A-7 Additional Considerations for Cluster Service
A-8 Windows Management Instrumentation
A-9 Provide MMC snap-in for management tools
A-10 Expose a COM-based scripting model
A-11 Use COM+ for Distributed Applications
A-12 Run without NetBIOS in Windows 2000-only environment

Appendix B Browser-Hosted Applications 74
Describes how requirements from the Desktop Application Specification apply to these types of clients.

Appendix C Usage of Non-Compliant 3rd Party Components 76

Appendix D Summary of Changes 77

Glossary 80

Chapter 1

Windows Fundamentals

Summary of Windows Fundamental Requirements

Rationale

Passing these requirements will help you to ensure that your application runs in a stable, reliable manner on Windows operating systems.

Customer Benefits

Customers can be confident that a compliant product will not adversely affect the reliability of the operating system.

Requirements

1. Perform primary functionality and maintain stability
2. Provide 32-bit components and document any 16-bit code
3. Support Long File Names and UNC paths
4. Support printers with long names and UNC paths
5. Do not read from or write to Win.ini, System.ini, Autoexec.bat or Config.sys
6. Ensure non-hidden files outside of your application directory have associated file-types, and all file-types have associated icons, descriptions and actions.
7. Perform Windows version checking correctly

8 Kernel mode drivers must pass verification testing

9 Hardware drivers must pass WHQL testing

10. Client components must comply with the Desktop Application Specification for Windows 2000

References

Supporting Long File Names:
http://msdn.microsoft.com/isapi/msdnlib.idc?theURL=/library/psdk/winbase/filesio_7qwj.htm

Using Driver Verifier tool:
http://www.microsoft.com/hwdev/driver/driververify.htm

General Functionality and Stability Test Procedure for the Certified for Microsoft Windows Logo
http://msdn.microsoft.com/certification/download.asp.

VeriTest-Rational Install Analyzer for Windows 2000
http://www.veritest.com/mslogos/windows2000

Link utility—Microsoft Platform SDK

How to Comply with Windows Fundamental Requirements

1. Perform primary functionality and maintain stability

Your application must perform its primary functionality without compromising the stability of the operating system or your application.

Primary functions are any functions so important that, in the estimation of a normal user, their inoperability or impairment would render the product unfit for its purpose. For example:

- Users should be able to enter new records, edit existing records, and delete existing records in a database application. They should be able to save the documents, open them later and print the documents on supported printers.
- Users should be able to view all supported resources within the expected scope of a resource management application. They should be able to create, save, open and print any supported reports.
- Users should be able to view expected data in an online analytical processing (OLAP) application and perform any supported analyses.
- Users should be able back up any supported files, folders, or drives on the local or supported remote machines from a server based backup application.

While users perform the primary functions of your application, it must not hang, crash or lose users' data. Primary functions should not become inoperable or obstructed, and the application must not disrupt Windows or any other applications running on the computer.

Users must be able to use system features supported by Windows 2000 with your application. For example:

- Your application should not crash, hang or lose data when supported users connect to or disconnect from the server hosting your application.
- A local user should be able to copy to and paste from the Windows clipboard in other applications while your application is running.

For more information on primary functions and stability, see the *General Functionality and Stability Test Procedure for the Certified for Microsoft Windows Logo*, located at http://msdn.microsoft.com/certification/download.asp.

2. Provide 32-bit components and document any 16-bit code

An application must be a 32-bit executable file of the Portable Executable (PE) format. This means that all executable files, including dynamic-link libraries (DLLs) and executable (EXE) program files, must be 32-bit files. You can test for the correct executable format by using the Link utility on the Microsoft Platform Software Developer's Kit (SDK) or the VeriTest-Rational Install Analyzer tool available for free download from http://www.veritest.com/mslogos/windows2000.

If your application is not represented in PE format, for example, interpreted code, then the "run-time engine" must be a Win32-based executable file in PE format. For example, if you develop an application in Microsoft Access, your application is an .mdb file, not an .exe file. However, msaccess.exe is a Win32-based executable file in the PE format.

Important: When you submit your product for compliance testing, you must list the filename and fully explain the use of any 16-bit file your application could install. This must be done in the electronic Vendor Questionnaire you submit with

your application.

Note: Each 16-bit file must be tested as part of the Certification process to ensure that system stability is not compromised. **This may require additional testing time and consequent additional charges by the testing agency**. The time and charges will be determined on a case-by-case basis by the testing agency.

Whenever possible, you should avoid using any 16-bit code, even for backward compatibility, because 64-bit versions of Windows will not support 16-bit code.

3. Support Long File Names and UNC paths

If your application exposes filenames to users and/or allows users to enter file names, then your application must support all valid Win32 file names, including Long File Names (LFNs) and Universal Naming Convention (UNC) names. LFNs and UNC names may be longer than the names allowed by 16-bit Windows and legacy DOS applications, and they may contain characters that were not allowed in 16-bit Windows or DOS.

For example, your application must properly recognize and display the following paths.

C:\documents and settings\joe user\my documents\my letter.txt

C:\documents and settings\Joe [joeuser]\my documents\Ca$h;flow\my letter to André.txt

D:\Our folder\project 10\project 11\project 12\project 13\project 14\project 15\ project 16\project 17\project 18\project 19\project 20\project 21\ project 22\project 23\project 24\project 25\project 26\project 27\ project 28\project 29\project 30\filename.ext

\\server\sharename\joe user\my letter.txt

All Win32 functions that create, open, locate and save files and folders use the constant MAX_PATH as the maximum buffer size for path information. Your application must allow users to create files with a total path length up to MAX_PATH, and your application must open any supported files the user creates, with your application and any other application, on paths up to MAX_PATH long. The MAX_PATH constant is defined in the platform SDK support file windef.h. In the current platform SDK, the value of MAX_PATH is “260.” You should use the MAX_PATH constant name instead of coding the value in your application. Using the name instead of the value will help you quickly adapt your application to future versions of Windows, which may support longer paths, but use the same constant name.

For more details on valid Win32 long file names, see the Platform SDK and http://msdn.microsoft.com/isapi/msdnlib.idc?theURL=/library/psdk/winbase/filesio_7qwj.htm

4. Support printers with long names and UNC paths

Windows allows long file names for Printers. If your application supports printing, it must accept names up to 220 characters long for printers, and print to devices with names such as these:

\\PrintServer44.domain.com\Duplex printer: number 0047 (accounting group)

Color laser - Research & Design Dept IP 123.456.78.9 see Fred or Wilma

Note: Commas and exclamation points are illegal printer name characters.

5. Do not read from or write to Win.ini, System.ini, Autoexec.bat or Config.sys

Your application must not read from or write to Win.ini, System.ini, Autoexec.bat,

or Config.sys. These file are not used by Windows 2000 systems and some users remove them.

6. Ensure non-hidden files outside of your application directory have associated file-types, and all file-types have associated icons, descriptions, and actions

Every non-hidden file that your application creates outside its directory in "Program Files" (see requirement 2.3) must have an associated registered file-type. This includes:

- Files created during installation
- Implementation and data files
- User created files that are native to your application

If the file-type is already registered, no action is required, though you may take over the file-type if you wish.

If the file-type is not already registered, you must do the following for each new file type:

- Provide an icon so that none of the files that your application creates is identified by the default Windows icon.
- Provide a friendly type description, for example, “Outlook Offline Mail File.”
- Ensure that each file-type has an associated action when double-clicked (e.g. launch your application and load the file), or is designated as “NoOpen.”

The NoOpen designation is for files that you don’t want users to open. When the user double-clicks a file marked as NoOpen, the operating system will automatically provide a message informing the user that the file should not be opened. Note that if an action is later associated with a NoOpen file type, the NoOpen designation will be ignored.

> **Exception**: If your application allows the user to save or export file types that aren’t native to your application, the user may choose to save a file as a type that has no association on the user’s computer. Your application may save the file as requested by the user, even though the file will have a default Windows icon.

Implementation details

To check if a file extension is already registered:

- Under HKCR, look for the key that is named with the 3 (or more) character file extension, such as ".txt". If the key with that file extension doesn't exist, you must register it (see below).

To register new file types:

- Create a file-type under HKCR. The file-type is the unique identifier for all files with a given file extension. For example, “txtfile” is the file-type for files with the .txt extension. Note that multiple extensions can point to the same file-type. For example, “.txt” and “.log” both point to the same file-type, “txtfile.” The default value for the file-type key is the “friendly name” that is displayed in explorer. For example, extensions that point to the “txtfile” key have the friendly name, “Text Document.”
- Under HKCR, create a three-character or longer extension for your file-type. We recommend 4 or 5 characters since this will help avoid file-type collisions and will make the file-type easier to identify. Then set the default value for the file extension to point to the file-type you've just created. For example the default value for ".txt" is the file-type "txtfile".

To register an icon for a file-type:

- Under the file-type key, create the 'DefaultIcon' key as a REG_SZ or REG_EXPAND_SZ and point it to an icon you provide. For example, for "txtfile" the value is "%SystemRoot%\system32\shell32.dll,-152" meaning "use the 152nd icon in shell32.dll".

To identify a file as “No Open”:

- Under the file-type key, add the “NoOpen” Reg_SZ value. Custom text can be added to the value if you would like to customize the message. See the “ocxfile” key in HKCR for an example.

Example: *Registry structure to associate “.txt” file extension with “Txtfile” file type:*

```
HKEY_CLASSES_ROOT\.txt
     (default) = "txtfile"

HKEY_CLASSES_ROOT\txtfile
        \DefaultIcon
           (default) = %SystemRoot%\system32\shell32.dll,-152
        \shell\open\command
           (default) = %SystemRoot%\system32\NOTEPAD.EXE %1
```

7. Perform Windows version checking correctly

Your application must verify that the operating system meets the minimum version requirements for your application. The application must also install and run on all later versions of that operating system.

For example, if the application requires Windows 2000 with Service Pack 1 (SP1), your version checking should allow installation on Major Version 5, Minor Version 0, SP1, and it must also install on all operating system version numbers that are greater than this number (such as Windows 2000 Service Pack 2 (SP2) and so on).

> **Exception:** In certain cases it is acceptable to block install on later versions of the Windows. If you choose to do this, you must:
>
> 1. Document this in the Vendor Questionnaire and explain the rationale.
> 2. Display a message to the user when blocking installation or execution that your application is not designed for the later Windows version.
>
> Examples where this is appropriate would be for low-level disk utilities. In this case, running such an application on a Windows version for which the product was not tested could potentially result in lost user data. For example, there could be changes in the file system of which the application was not aware.

For applications that will run only on Windows 2000 and later versions, we recommend that you use the VerifyVersionInfo API to determine the version of the operating system. Note however that this API is not available on down-level platforms. For applications that also run on Windows NT 4, you should use the **GetVersionEx()** API to determine the OS version. See the code samples below for examples of each implementation.

Code sample using VerifyVersionInfo (for Windows 2000 and later)

Using VerifyVersionInfo helps applications more reliably determine if the application can be installed on a particular OS. In the past, the application would request the version info from the OS and then write code to decide it the current version met its needs. In contrast, with VerifyVersionInfo, the application tells the OS what version it needs, and then asks the OS if the current version meets this. Fewer lines of code are needed.

```
#ifdef Windows2000
//
//This sample is for Windows2000 and later versions
//
BOOL bIsWindowsVersionOK(DWORD dwMajor, DWORD dwMinor, DWORD wSPMajor )
    {

    OSVERSIONINFOEX osvi;
    DWORDLONG dwlConditionMask;
  ZeroMemory(&osvi, sizeof(OSVERSIONINFOEX));

  osvi.dwOSVersionInfoSize = sizeof(OSVERSIONINFOEX);
  osvi.dwMajorVersion = dwMajor;
  osvi.dwMinorVersion = dwMinor;
  osvi.wServicePackMajor = wSPMajor;

  // Set up the condition mask.

  VER_SET_CONDITION( dwlConditionMask, VER_MAJORVERSION,
VER_GREATER_EQUAL );
  VER_SET_CONDITION( dwlConditionMask, VER_MINORVERSION,
VER_GREATER_EQUAL );
  VER_SET_CONDITION( dwlConditionMask, VER_SERVICEPACKMAJOR,
VER_GREATER_EQUAL );

  // Perform the test.
  return VerifyVersionInfo(&osvi,
                           VER_MAJORVERSION | VER_MINORVERSION |
VER_SERVICEPACKMAJOR,
                       dwlConditionMask);
    }
#endif
```

Code sample to verify the Windows version on down-level platforms

This code runs on all 32-bit Windows platforms. Note, if you need to verify the version of an NT4 Service Pack prior to Service Pack 4, you should also query the following registry key to determine the SP level, as in this sample.

```
HKLM\system\CurrentControlSet\control\windows\CSDVersion
```

The values of CSDVersion will be 0x100 for Service Pack 1, 0x200 for Service Pack 2, etc. See below for code sample to determine the Service Pack level.

```
BOOL bIsWindowsVersionOK(DWORD dwMajor, DWORD dwMinor, DWORD dwSPMajor )
{
OSVERSIONINFO osvi;

// Initialize the OSVERSIONINFO structure.
//
ZeroMemory(&osvi, sizeof(OSVERSIONINFO));
osvi.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);
GetVersionEx((OSVERSIONINFO*)&osvi);

// First the major version
if ( osvi.dwMajorVersion > dwMajor )
   return TRUE;
else if ( osvi.dwMajorVersion == dwMajor )
   {
   // Then the minor
   if (osvi.dwMinorVersion > dwMinor )
      return TRUE;
   else if (osvi.dwMinorVersion == dwMinor )
      {
      // OK, better check the Service Pack
      if ( dwSPMajor &&
      osvi.dwPlatformId == VER_PLATFORM_WIN32_NT )
         {
         HKEY  hKey;
```

```
            DWORD dwCSDVersion;
            DWORD dwSize;
            BOOL  fMeetsSPRequirement = FALSE;

            if (RegOpenKeyEx(HKEY_LOCAL_MACHINE,
               System\\CurrentControlSet\\Control\\Windows",
               0, KEY_QUERY_VALUE, &hKey) == ERROR_SUCCESS)
               {
               dwSize = sizeof(dwCSDVersion);
               if (RegQueryValueEx(hKey, "CSDVersion",
                  NULL, NULL,
                  (unsigned char*)&dwCSDVersion,
                  &dwSize) == ERROR_SUCCESS)
                  {
                  fMeetsSPRequirement =
                     (LOWORD(dwCSDVersion) >= dwSPMajor);
                  }
               RegCloseKey(hKey);
               }
               return fMeetsSPRequirement;
               }
            return TRUE;
            }
         }
      return FALSE;
      }
```

8. Kernel mode drivers must pass verification testing

Poorly written kernel mode drivers have the potential to crash the system. Therefore, it is critical that any application that includes kernel mode drivers, such as anti-virus products, be thoroughly tested to minimize this risk.

If your application includes any kernel mode drivers, each of these drivers must pass validation triggered by the Windows 2000 Driver Verifier Manager tool (Verifier.exe). Driver Verifier is located in the \system32 folder on Windows 2000 systems. See http://www.microsoft.com/hwdev/driver/driververify.htm for more information on using the Driver Verifier Manager and diagnosing driver problems.

9. Hardware drivers must pass WHQL testing

If your product includes drivers for hardware devices, these drivers must be tested by Windows Hardware Quality Labs (WHQL). Please note that turn-around times at WHQL can be up to 30 days. For more information, see http://www.microsoft.com/hwdev/winlogo.

10. Client components must comply with the Desktop Application Specification for Windows 2000

Any client components that run on Windows 2000 Professional and that ship with your application must comply with the Desktop Application Specification for Windows 2000 in order for the whole application to be Certified for Windows. Distributed applications that have client components are eligible to carry both Windows 2000 Server and Windows 2000 Professional designations on their Certified for Windows logo.

The following clients may be exempted from complying with the Desktop Application Specification:

- Administration tools for your server application are temporarily exempt thru March 31, 2001. After that date, they will be required to comply with the Desktop Application Specification.
- “Agents” that have no UI

In addition, special considerations apply for browser-hosted client applications.

Please see Appendix B for details.

Note: Testing for client components will be performed on Windows 2000 Professional, but not on down-level operating systems.

How to Pretest Applications for Windows Fundamental Requirements

How to pretest that your application is 32-bit.

Use the VeriTest-Rational Install Analyzer for Windows 2000, available from http://www.veritest.com/mslogos/windows2000, OR you can use the Link utility in the Microsoft Platform SDK.

If you use the Link utility, at the command line, type: “link –dump –headers exename.exe | more“. If your application is in the proper PE format, the output of this command will be:

```
Microsoft (R) COFF Binary File Dumper Version 5.12.8181
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.

Dump of file exename.exe

PE signature found

File Type: EXECUTABLE IMAGE
.
.
.
```

You should *not* get the following output, which would be produced if the application were 16-bit:

```
Microsoft (R) COFF Binary File Dumper Version 5.12.8181
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.

Dump of file exename.exe
LINK : warning LNK4095: " exename.exe" is an NE format executable; use
EXEHDR to dump it
.
.
.
```

To pretest for entries in Win.ini, System.ini, Autoexec.bat, and Config.sys files:

1. Create and save copies of these files before installing the application on a computer where Windows has been freshly installed.
2. Install the application.
3. Launch the application, exercise basic functionality, then close the application.
4. Compare the installed version of the files to the saved version of the files, verifying that no changes have been made. The Install Analyzer will verify this for you.

To test whether your product properly handles Long File Names:

LFNs must do the following:

- Allow plus signs, periods, commas, semicolons, equal signs, and square brackets anywhere.
- Not save leading or trailing spaces. You can test for this by typing “###test###” or similar text in the “Save As” dialog box. (In this section, the number sign (#) indicates a spacebar space. ANSI 0032) The program should strip the spaces and add an extension, returning the file name “test.ext”.

  Note: The ANSI 0160 character, used as a non-breaking space character in many fonts, should always be retained, even if used as the leading or trailing character in folder and file names.
- Support MAX_PATH characters (including the path and extension).

- Test the following list of file names (not including the quotes), which should save to the hard disk as indicated:

If you type…	*It should be saved as*
"test"	"test.ext"
" test"	"test.ext"
"test "	"test.ext"
"test#;#+#,#=#[#]"	"test#;#+#,#=#[#].ext"
" test " (non-breaking space characters ANSI 0160)	" test .ext"
"…..test….."	"…..test……ext"
"\\server\share one\folder three\file"	"\\server\share one\folder three\file.ext"

NOTE: if you are typing a UNC name that contains spaces from a command line, you will need to enclose the name in quotes.

To run verification tests on kernel mode drivers

1. Connect a kernel mode debugger to your test machine. Several suitable debuggers are available, including character mode and GUI debuggers, on the Windows 2000 SDK and DDK CDs, and on the Windows 2000 Symbols and Support CD.
2. Install your application with its kernel mode drivers.
3. Start Verifier.exe and select the Settings tab. Look for each driver's name in the Driver list. Select each of your application's drivers on the list and click Verify. If you don't find one or more of the drivers' names in the Driver list, enter the names in the "Verify these additional drivers after next reboot" edit box. Separate multiple driver names in the edit box with spaces and use only the name of the driver and its extension. Do not include path information.
4. Check only these options in "Verification type":
   a. Special pool
   b. Force IRQL checking
   c. Pool tracking
   d. I/O verification
5. Click Apply, Exit and reboot.
6. After reboot, start your application and run a series of typical user tests. If the kernel detects any errors in your driver during boot or during the user tests, it will halt Windows 2000 and display the appropriate information on the character mode screen (blue screen) and on the kernel debugger. If pool corruption is indicated in the debugger, uncheck "Pool tracking" in "Verification type" and restart. When you've eliminated the errors, re-enable "Pool tracking" and retest.
7. If you find no errors, restart Verifier.exe. Select each driver name again and click Verify, or enter the drivers' names (filename.extension, without path) again in the "Verify these additional drivers after next reboot" edit box, separated by spaces.
8. Uncheck all options in "Verification type", then check "Low resources simulation."
9. Click Apply, then exit and reboot. Start your application and run a series of typical user tests.
10. The low resources simulation option causes the Kernel to return invalid data

and error codes to your driver periodically. As you run user tests, your driver should handle the invalid codes gracefully, perhaps by declining user requests or posting error dialogs. If your driver tries to use the invalid data or ignores error codes, your application will become unstable and fail. The exact symptoms vary with the application and purpose of the driver. The goal here is to ensure that your driver does not crash or hang the system. It is acceptable for applications using your driver to experience intermittent failures, but the system must NOT crash.

Chapter 2

Install/Uninstall

Summary of Install/Uninstall Requirements

Rationale

Install and uninstall issues are some of the most common sources of application interoperability problems. These requirements help to ensure that the user has successful installation and uninstallation experiences, and that the application co-exists in a friendly way with other applications.

NOTE: It is HIGHLY recommended that you use the Windows Installer service for the installation of your application. The Windows Installer service is an operating system component that centrally manages application installation configuration, as well as application uninstall. Using the Windows Installer service allows the operating system to manage application setup and configuration, which enables the following:

- Management of reference counting and version checking of shared components, which helps ensure that applications better co-exist with one another.
- Self-repair of damaged applications at runtime: When the application is launched, the Windows Installer will check to ensure that the application is properly installed, and if it is not, will automatically repair the application on-the-fly.
- Transacted install: In the event that the installation is not completed (for example, if there is a network failure), the Windows Installer can roll back to the earlier installed version of the application without error.
- Reliable and complete uninstall, including correct handling of shared components.
- Ability to perform installation on secure systems (for non-administrators and non-power users).
- Application installation more readily supports mass deployment in organizations using Windows 2000 software management, and as a result of standard support for customization and unattended install.

The Windows Installer service enables all of this functionality using packages that describe application configurations. The Windows Installer ships in Windows 2000 and is also available for redistribution on Windows NT 4, Windows 98, and Windows 95.

For more information, see the Windows Installer Programmer's reference in the Microsoft Platform SDK.

Customer Benefits

- Fewer problems arising from application install: installation and uninstallation of applications is less likely to affect the functioning of another application because reference counting and version checking are properly implemented.
- Customers can properly uninstall your application.

Requirements

1. Do not attempt to replace files that are protected by Windows File Protection

2. Do not overwrite non-proprietary files with older versions
3. Install to Program Files by default
4. Install shared files to the correct locations
5. Refcount all shared application files during installation
6. Support Add/Remove Programs properly
7. Decrement the count on shared application files during uninstall
8. Ensure correct uninstall

How to Comply with Install/Uninstall Requirements

NOTE: If your application uses the Windows Installer, the install/uninstall requirements from Chapter 2 of the Desktop Application Specification supercede requirements 2.2 thru 2.8 in this chapter, with the exception that Desktop Requirement 2.6 (support Advertising) is not required for server applications.

1. Do not attempt to replace files that are protected by Windows File Protection

Your application must not attempt to replace any files that are protected by Windows File Protection (WFP). To ensure that you do not invoke WFP, call **SfcIsFileProtected** when installing any file you did not create. The Windows Installer service version 1.1 automatically does this for you.

Protected files include the following files that ship on the Windows 2000 CD:

- Most .SYS, .DLL, .EXE and .OCX files.
- The following fonts: micross.ttf, tahoma.ttf, tahomabd.ttf; fixedsys.fon, dosapp.fon, modern.fon, script.fon and vgaoem.fon

Note: Some redistributable files, such as specific versions of Microsoft Foundation Classes (MFC) DLLs, are installed by Windows 2000 and are protected by WFP.

Protected files form the core of the operating system and it is essential for system stability that the proper versions be maintained. These files can only be updated via Service Packs, operating system upgrades, QFE hotfixes, and Windows Update. Applications cannot replace them, and attempting to replace these files by means other than those listed above will result in the files being restored by the Windows File Protection feature (see below).

If your application requires newer versions of these components, your application must update these components using an approved Microsoft Service Pack that installs the required versions.

About Windows File Protection:

Windows File Protection is a feature of Windows 2000 that prevents the replacement of essential system files. WFP runs as a background process on Windows 2000 and monitors the files listed in the preceding section. When WFP detects that a protected file has been changed, it restores the original.

The following code shows how to check if a file (in this case “ntdll.dll”) is protected by WFP. Note that **SfcIsFileProtected** is Unicode-only and requires a fully qualified path.

```
SHGetFolderPath(NULL,CSIDL_SYSTEM, NULL, 0, szSystemDir);
PathAppend(szSystemDir,"ntdll.dll");
MultiByteToWideChar(CP_ACP, 0, szSystemDir, -1, wzFileName, 265);

if ( SfcIsFileProtected(wzFileName) )
  MessageBox(hWnd,szProtected,szSystemDir,MB_OK);
else
  MessageBox(hWnd,szNotProtected,szSystemDir,MB_OK);
```

2. Do not overwrite non-proprietary files with older versions

You application's install must properly version check to ensure the latest file versions are installed. Installing an application must never regress any files that you do not produce or are shared by applications you don't produce.

The **VerInstall** or **GetFileVersionInfo** APIs are recommended for performing the version check. **VerInstall** is easiest because it uses **GetFileVersionInfo** internally and simplifies the process. Below is a simplified example for how to use **VerInstall**:

```
TCHAR szOldDir[MAX_PATH], szTempDir[MAX_PATH];
UINT cchTemp = MAX_PATH;
DWORD dwResult = VerInstallFile(0, TEXT("file.dll"), TEXT("file.dll"),
g_tszSetupDirectory, g_tszInstallDirectory, szOldDir, szTempDir,
&cchTemp);
if (dwResult != 0) {
   Error(dwResult);
}
```

3. Install to Program Files by default

By default, your application must install non-shared components into an appropriate subdirectory where the user's program files are stored (e.g. Program Files). This folder is represented by CSIDL_PROGRAM_FILES. On English systems, this folder is often "C:\Program Files". However, do **not** hardcode that path, as it is not universal.

The recommended way to locate this folder by using **SHGetFolderPath** (CSIDL_PROGRAM_FILES,…). If you are using the Windows Installer, this folder is represented by the `ProgramFilesFolder` property in the Windows Installer-based package.

Exceptions:

If you are upgrading a previously installed version of your application, it is acceptable to default to the directory where that version exists.

Exceptions to this requirement may be allowed for applications that prefer a dedicated drive for installation.

Considerations for shared components

In some cases, shared components may need to be placed in locations other than the application directory. See next requirement.

4. Install shared files to the correct locations

The proper location for shared components depends on whether these components are shared across companies or by a single company.

- Shared components that are private to a single software vendor must be installed in one of two places. Do not store these files in the system directory.

  common files directory\<company name>

  %ProgramFiles%\<company name>\Shared Files

The common files directory can be accessed by passing CSIDL_PROGRAM_FILES_COMMON to the **SHGetFolderPath** API.

- Non side-by-side OCXs and DLLs that are shared by multiple software vendors can be placed in the system directory to ensure backward compatibility with those applications.

  You must document in your Vendor Questionnaire any cases where your software application writes to the system directory.

- New control panel applets (CPLs) must be installed in the application directory on Windows 2000. Register the path by adding a value under either of the following registry keys:

  ```
  HKLM\software\microsoft\windows\CurrentVersion\control panel\cpls
  HKCU\software\microsoft\windows\CurrentVersion\control panel\cpls
  ```

  Example of a name / value pair under this key:

  ```
  MyCpl = "%ProgramFiles%\MyCorp\MyApp\MyCpl.cpl"
  ```

- Services and device drivers should be placed in the system directory.

Note: The system directory is not locked down when the user is a power user or administrator, so legacy components or globally shared components can still be placed there. However, Windows File Protection prevents the replacement of protected operating system files. See requirement #2.

5. Refcount all shared application files during installation

If your application does not use the Windows Installer, the application installer must refcount *all* shared components. This must be done under the following registry key:

```
[HKEY_LOCAL_MACHINE]\SOFTWARE\Microsoft\Windows\Current Version\SharedDLLs
```

If your installer finds a shared component already on the system that is not registered, the SharedDLL count should be incremented by 1 plus the number of components in your application that consume this component. For example, if your application installs three components that use a shared file and the shared file is already registered, your installer will bump the SharedDLL count by 3. However, if the shared file was already on the system but no SharedDLL exists for it (that is, a previous installer did not create the refcount), then set the SharedDLL count to 4. That way, when your application is uninstalled, it leaves the shared file on the system with a refcount of 1.

Components that are shared with applications that don't use Windows Installer service need to properly implement the DLLRegisterServer and DLLUnregisterServer entry points if they need to do any registration at install time. (DLLs which use the Windows Installer should use the registration services provided by the Windows Installer.)

6. Support Add/Remove Programs properly

Your application must supply all the information in the following table so that Add/Remove Programs in the Control Panel can obtain information about the application as needed. You can write this information directly to the registry during install, or if you are using the Windows Installer service, you can set these values using properties in the Windows Installer-based package.

The registry values should be written under the following key:

```
HKEY_LOCAL_MACHINE
 \Software
  \Microsoft
```

```
\Windows
 \CurrentVersion
  \Uninstall
   \{ProductCode}
```

Registry value	Type	Corresponding Windows Installer Property	Contains
DisplayName	REG_SZ	ProductName	Display name of application.
UninstallPath	REG_EXPAND_SZ	N/A	Full path to the application's uninstall program.
InstallLocation	REG_EXPAND_SZ	ARPINSTALLLOCATION	Full path where application is located (folder or .exe).
Publisher	REG_SZ	Manufacturer	Publisher/Developer of application.
VersionMajor	DWORD	ProductVersion	Major version number of application.
VersionMinor	DWORD	ProductVersion	Minor version of application.

Note: Property names are case-sensitive.

You can also provide additional properties to present in Add/Remove Programs if you like, such as product ID, online support information, etc. See the platform SDK for full details.

7. Decrement the count on shared application files during uninstall

The uninstaller must accurately decrement the count on all the components your application uses that are installed as shared components. Delete them when the refcount reaches zero.

8. Ensure correct uninstall

Your application must provide an automated uninstaller that removes the application. The uninstaller must be properly registered as specified in the requirement 2.7, and it must remove the following:

- All non-shared application files and folders
- Shared application files whose refcount reaches zero
- Registry entries, except for keys that might be shared by other programs
- All shortcuts from the Start Menu that the application created at install-time
- The uninstaller itself

The following should remain on the hard disk:

- User data files.
- Shared application files that have a non-zero refcount.
- Other resources that other programs use, sharable fonts, and sharable registry entries.

You must explain in the Vendor Questionnaire everything you leave behind.

Tip: If your application creates temporary files that should be removed during uninstallation, create a zero-length (0) file with the same name at installation time. Examples of such files would be .gid files created by Help.

NOTE: If your application uses the Windows Installer service and follows componentization rules and uses only native Windows Installer actions to modify the computer, uninstall capability is provided automatically.

How to Pretest Applications for Install/Uninstall Requirements

To pretest uninstallation:

1. Take a snapshot of a computer's directory tree and registry, install the application, uninstall the application, and take another snapshot.
2. Verify that the snapshots before the install and after the uninstall are the same, except for the following which should be left on the machine: user created files, and shared components that are still required by other programs.

The VeriTest-Rational Install Analyzer for Windows 2000, available at http://www.veritest.com/mslogos/windows2000, can help you do this.

To pretest interoperability with other applications that share components:

To ensure that your application interoperates well with other applications (that do not use the Windows Installer) that share some of the same components that your application uses, you should test and verify each of the following scenarios:

	Scenario 1	***Scenario 2***	***Scenario 3***	***Scenario 4***
Step 1	Install your app	Install your app	Install "other" app	Install "other" app
Step 2	Install "other" app	Install "other" app	Install your app	Install your app
Step 3	Uninstall your app	Uninstall "other" app	Uninstall "other" app	Uninstall your app
Step 4	Verify "other" app still works	Verify your app still works	Verify your app still works	Verify "other" app still works
Step 5	Uninstall "other" app	Uninstall your app	Uninstall your app	Uninstall "other" app
Step 6	Verify that shared components are deleted	Verify that shared components are deleted	Verify that shared components are deleted	Verify that shared components are deleted

Chapter 3

User Interface Fundamentals

Summary of Requirements for UI Fundamentals

NOTE: It is not required that a server application have any graphical UI. However, any graphical UI that is presented must meet these requirements. Regardless of whether any graphical UI is presented, all applications must adhere to requirement 3.5.

Rationale

- Consistency and accessibility among Windows-based applications increases customer confidence in Windows applications and the Windows platform.
- Meeting UI requirements enables the use of sophisticated automation tools, including testing tools, task automation tools such as intelligent agents, and new input methods such as voice input.
- Organizations that use the Windows platforms require software to be usable by a wide range of users in order to comply with employment and equal rights legislation.
- Meeting these requirements helps you ensure that software will be compatible with future enhancements that are planned for the Windows platform, including speech input, speech output, and intelligent task automation.

Customer Benefits

- Providing a consistent user interface reduces training, support, and testing costs for customers.
- Meeting UI requirements enables your business customers who are located in the United States to comply with the Americans with Disabilities Act, which requires that they provide reasonable accommodation for employees with disabilities. It also enables government agencies, schools, and organizations receiving federal funding to comply with section 508 of the Rehabilitation Act, which was strengthened in the reauthorization of 1998. Similar legislation exists in a number of states and countries, and is pending in many more.
- Meeting UI requirements enables your customers to purchase software that meets standards for usability, such as HFES/ANSI 200 and ISO 9241. Such standards will continue to expand to incorporate high-level requirements for usability and accessibility.
- Meeting UI requirements enables over 49 million people in the United States who have disabilities to use your products. The U.S. Access Board estimates that one in five people, and one in eight Internet users, have some form of functional limitation.
- Meeting UI requirements enables organizations to benefit by keeping valuable

employees, even when they receive a permanent or temporary disability, such as repetitive strain injury associated with typing or using a mouse, or experience functional limitations that are a natural part of aging.

Requirements

1. Support standard system size, color, font, and input settings
2. Ensure compatibility with the High Contrast option
3. Provide documented keyboard access to all features
4. Expose the location of the keyboard focus
5. Do not place shortcuts to documents, help or uninstall in the Start Menu

References

Guidelines for accessibile software design: http://www.microsoft.com/enable
Microsoft Active Accessibility: See the Platform SDK
Microsoft media player information:
http://www.microsoft.com/windows/mediaplayer

Microsoft Windows Guidelines for User Interface Design.
Available from MSDN and MS Press.

How to Comply with Requirements for UI Fundamentals

1. Support standard system size, color, font, and input settings

Your application must read and use system-wide user interface settings when displaying customized controls or window content. **Standard controls provided by USER32.DLL and COMCTL32.DLL automatically support all of the required settings.** Applications can add support for these settings to their own windows by calling DefWindowProc in the window procedure. Applications should be careful to handle these settings when:

- Creating custom controls
- Creating owner-drawn controls
- Superclassing or subclassing to alter the normal standard control behavior
- Handling a message and not calling DefWindowProc (especially when drawing any non-client areas)
- Handling low-level input that bypasses normal mouse and keyboard messages (such as double-click and shift-state detection)

These settings are queried using the **GetSysColor, GetSystemMetrics**, and **SystemParametersInfo** functions. For more information, see the "System Information Functions" in the Microsoft Platform SDK, or at

http://msdn.microsoft.com/library/psdk/sysmgmt/sysinfo_8stv.htm

All applications must support:

SPI_GETHIGHCONTRAST	All **GetSysColor** settings are required for menus, dialog boxes, and other standard UI elements. *See Requirement 2 in this chapter*.
SPI_GETWORKAREA	Primary display monitor work area size

Customized fields for text selection or editing must support:

SPI_GETCARETWIDTH	Caret width in edit field

Customized combo boxes must support:

SPI_GETCOMBOBOXANIMATION	Slide-open effect for combo boxes must be disabled when this setting is false

Customized keyboard handlers (low level input) must support:

SPI_GETKEYBOARDDELAY	Keyboard repeat-delay setting
SPI_GETKEYBOARDSPEED	Keyboard repeat-speed setting
SPI_GETFILTERKEYS	Expanded range for above

Customized menus must support:

SM_CYMENUCHECK, CXMENUCHECK	Default menu check-mark dimensions
SM_CYMENU	Single-line menu bar height
SPI_GETNONCLIENTMETRICS	
lfMenuHeight	Menu bar height
lfMenuFont	Menu bar font
lfMessageFont	Message box font
SPI_GETSELECTIONFADE	Menu fade must be disabled when this setting is FALSE
SPI_GETMENUANIMATION	Menu animation must be disabled when this setting is FALSE
SPI_GETMENUFADE	Menu fade animation must be disabled when this setting is FALSE

Specialized mouse functions must support:

SM_CYDOUBLECLK, SM_CXDOUBLECLK	Maximum vertical and horizontal distances the mouse can move between successive clicks and still be interpreted as a double-click
SM_CYDRAG, SM_CXDRAG	Minimum vertical and horizontal distances the mouse must be moved with button down to start a drag operation
SPI_GETMOUSEHOVERHEIGHT SPI_GETMOUSEHOVERWIDTH	Height and width of the rectangle within which the mouse pointer has to stay to activate hover effects
SPI_GETMOUSEHOVERTIME	Mouse hover time

Customized scroll bars must support:

SM_CYHSCROLL	Height of horizontal scroll bar
SM_CXVSCROLL	Width of a vertical scroll bar

Application sounds must support:

SPI_GETSHOWSOUNDS	When TRUE, the application must present all information visually rather than by sound alone

Customized tooltips and status bars must support:

SPI_GETNONCLIENTMETRICS	
SPI_lfStatusFont	Font used in status bars and tooltips
SPI_GETTOOLTIPANIMATION	ToolTip animation must be disabled when this setting is FALSE

Customized window frames must support:

SM_CYSMCAPTION	Small caption height
SPI_GETNONCLIENTMETRICS	
lfCaptionFont	Caption font information
SM_CYBORDER, SM_CXBORDER	Minimum thickness of a line, in pixels. Applications should not draw lines thinner than this because on high-resolution monitors a 1-pixel line might be nearly invisible. (The display driver sets this number, and it is also increased when the user selects the Custom System Font Size in Control Panel.)
SM_CXEDGE SM_CYEDGE	Thickness of a 3D edge, in pixels. Each 3D edge is comprised of two adjacent lines, so the thickness is 2 * SM_CxBORDER.
SM_CXSIZEFRAME SM_CYSIZEFRAME	Thickness of the complete resizable window frame, in pixels.
SM_CYFIXEDFRAME SM_CXFIXEDFRAME	Thickness of a non-resizable window frame, in pixels. This is made up of one 3D edge whose thickness is SM_CxEDGE plus one line of space whose thickness is SM_CxBORDER.

2. Ensure compatibility with the High Contrast option

Your application must be compatible with the High Contrast option, which

indicates that the user requires a high degree of contrast to improve screen legibility. **Standard controls provided by USER32.DLL and COMCTL32.DLL automatically support all of the required settings.** Applications only need to support it explicitly when creating custom window classes or controls or altering the normal behavior of standard windows or controls.

Applications can determine the value of the High Contrast option using the **SystemParametersInfo** function with the SPI_GETHIGHCONTRAST constant. When this option is set, your application must:

- Display all menus and dialog boxes using the color scheme returned by **GetSysColor** function. This also applies to any other UI elements that are required to adjust colors in the application's UI.
- Allow the user to adjust the colors used to display everything in the application's windows.
    - This must be possible using display options that override the colors normally used by the application or specified in the document, and must not alter the content of the document or affect other users.
    - The preferred method is to use the corresponding colors returned by the GetSysColor function, but the application can provide its own display options.
    - Always draw foreground objects in colors designated as foreground colors and fill backgrounds with the corresponding background colors. This is required whether colors are selected using the GetSysColor function or through the application's own options. For example, anything drawn using window text color (COLOR_WINDOWTEXT) must be drawn on the window background color (COLOR_WINDOW), and anything drawn using highlight text color (COLOR_HIGHLIGHTTEXT) must be drawn on highlight background (COLOR_HIGHLIGHT).
- Make sure any information normally conveyed by color is available through other means, such as sound or a visual display, because the user may have chosen a monochrome appearance scheme.
- Omit any images or patterns drawn behind text, and make sure any important information conveyed with such backgrounds is available through other means.

Note: Users can adjust the High Contrast option from the Accessibility Options section of Control Panel, by selecting the Display tab and then the Use High Contrast check box.

Users can adjust the values returned by **GetSysColors** using the Display section of Control Panel, by selecting the Appearance tab.

The High Contrast option can be used with any appearance scheme, and selecting a scheme through the Display section of Control Panel does not affect the High Contrast option setting.

Exception: Certain application features may be exempted from High Contrast requirements when the use of color is intrinsic and indispensable to the goal of the feature. Examples include:

- Palettes or swatches where the user selects from a range of displayed colors. In this case, the application can display the color but must provide a text description such as a name (light blue) or numeric value (RGB 0, 255, 255).
- Animation, video, and graphic images when the information content is available through other means.

When the major features in the application support the High Contrast option, exceptions may also be made for minor features that are not required for the operation of the program.

Requests for exemptions are judged on a case-by-case basis. You must detail these exceptions in your Vendor Questionnaire.

3. Provide documented keyboard access to all features

Your application must provide keyboard access to all features, so that a mouse or other pointing device is not required for its use. **Standard controls provided by USER32.DLL and COMCTL32.DLL automatically support all of the required settings.** Applications must explicitly provide this support only when:

- Creating custom window classes or controls
- Altering the normal behavior of standard windows or controls
- Assigning keyboard navigation in dialog boxes.

User interface elements that rely on standard Windows keyboard documentation should support all equivalent standard window class keyboard behavior. If a specific user interface element, such as a custom control, requires a mouse or other pointing device, its functionality must also be available through keyboard input.

All keyboard techniques required to operate the application must be documented, except for elements that follow Windows conventions, e.g., standard menus and controls. Applications must document non-intuitive complex procedures, customized window class usage, and controls that do not conform to normal conventions. The product's standard documentation must either include this information or direct the user to the appropriate documents.

Example: Information can be delivered by online help, with text file on the product CD, or through the product's Web site.

Note: Keyboard behavior of standard window classes is documented in *The Microsoft Windows Keyboard Guide*, which is available on http://www.microsoft.com/enable.

Exception: Exemptions may be made in the following cases:

- Applications that rely on specialized input devices, such as graphing tablets.
- Situations where the mouse targets are no larger than a pixel. For example, painting with the mouse. These features may rely on the MouseKeys feature built into the 32-bit Microsoft Windows operating systems to allow users to move the mouse pointer using the keyboard. However, this is not acceptable for drawing when the user can independently manipulate separate text and graphic objects. This exemption applies to individual features within a product, not to the entire application.

When the major features in the application have keyboard access, exceptions may be made for minor features that are not required for the operation of the program.

Requests for exemptions are judged on a case-by-case basis. You must detail these exceptions in your Vendor Questionnaire.

4. Expose the location of the keyboard focus

Your application must visually indicate the location of the keyboard focus, and notify other software of this location by using Microsoft Active Accessibility™ or by moving the system caret.

The application must display a visual focus indicator at all times when it is the active window so users can anticipate the effects of keystrokes they type.

Exposing its location to other software enables the use of the panning software supported by many display adapters and accessibility aids such as the Magnifier accessory included with Windows 2000.

When it is not feasible to use Microsoft Active Accessibility, the application may indicate the focus location by moving the system caret. The caret is normally the blinking vertical bar that the user sees when editing text, but it can be placed anywhere on the screen, made any shape or size, and even made invisible. If it is invisible, it can be moved to indicate the focus location to applications without disturbing what the user sees on the screen.

To create an invisible caret call the **CreateCaret** function to set the caret's size and shape and the **SetCaretPos** function to move it to wherever you are drawing the visual focus indicator (the highlighted cell, icon, button, and so on). Note that it is present but remains invisible unless you explicitly make it visible.

Each time the focus moves to a new object with a different size, the application should call **DestroyCaret** and then use **CreateCaret** to specify the size of the new object. The system caret must cover the bounding rectangle of the screen element, so that screen magnification tools can allow the user to zoom-in on different portions of the region.

An application should only display focus and selection indicators when they are in the active window. When the window loses activation, the application should remove the visual indicator and call the **DestroyCaret** function.

> **Exception:** Applications and features that are exempt from the keyboard access requirement as described earlier in this section are also exempt from the requirement to expose keyboard focus location. However, it is strongly recommended that applications expose the keyboard focus location for any feature that provides keyboard access.

5. Do not place shortcuts to documents, help, or uninstall in the Start Menu

The Start Menu is designed to give users easy access to launch applications. Usability studies show that when the Start Menu becomes too cluttered, users can no longer do this.

- Do not place shortcuts to documents, such as readme files, in the Start Menu. If you have important information that the user should see, consider displaying that information during the install process.
- Do not put shortcuts to help files in the Start Menu. Users can access help once they launch the application.
- Do not place shortcuts to uninstall in the Start Menu. The Add/Remove Program control panel applet provides this functionality.

The following behaviors, though not required, are recommended:

- Place your icon to launch your application directly under Start -> Programs. Avoid placing it in a folder under programs. In particular, do not create a folder in the Start Menu in which you only put one item. Often, applications will create a folder based on Company name and then put a single shortcut to launch the application inside that folder. Instead, consider renaming the shortcut to include the company name and dropping the use of the folder.

Programs →	My Company →	My App	(Avoid this)
Programs →	My Company My App		(Recommended)

- Do not put anything in the top of the Start Menu, as users consider this their own personal space.
- If you have support applications, tools, or utilities associated with your application, and you wish to publish these in the Start Menu, create a single folder in the Start Menu as a peer of the icon to launch your application and place them there.

How to Pretest Applications for UI Fundamentals

To pretest system metrics support:

1. Open Control Panel, and then open the Display applet.
2. Choose the Appearance tab.
3. Select a scheme with the "(extra large)" designation, such as Windows Standard (extra large) scheme. Click on Apply.
4. Test the application's main screens, dialog boxes, and controls to ensure that these changes have been instated and the application is still usable.

   Screen elements should be displayed correctly and compatible with the appearance scheme, i.e., menu items, title bars, icons and icon titles, window borders and scroll bars are displayed with appropriate sizes and fonts. Text and controls should never overlap and text should not be truncated inappropriately. Examples of appropriate truncation are

   - Text strings too large to be displayed in a column in a list view control. The beginning of the text is displayed, followed by "…" to indicate the truncation.
   - Text that runs beyond the frame of a window, but can be viewed using the horizontal or vertical scroll bar.

To pretest high-contrast support:

1. Open Control Panel, and then open the Accessibility Options.
2. Choose the Display tab.
3. Select the Use High Contrast option.
4. Choose Settings.
5. Choose Custom and verify that High Contrast Black is selected. Apply these settings.
6. Go back into your product and test the major screens, dialog boxes, and controls to ensure that these changes have been properly reflected.
7. Repeat, selecting the High Contrast White option in step 5.

   The application should still be usable. Screen elements should be displayed correctly and be compatible with the appearance scheme. For example, make sure that text contrasts with its background, that graphics on toolbar buttons are distinguishable, that lines are still visible, and that no images or complex backgrounds appear behind text.

   If anything within the window is not displayed using the selected color scheme, verify that the user can adjust the display colors without having to

alter the document.

To pretest documented keyboard access:

1. Open Control Panel, and then open the Accessibility Options.
2. Choose the Keyboard tab.
3. Select the Show Extra Keyboard Help. Apply these settings.
4. Exercise the major functions within your application without using a mouse.
5. Verify that a mouse is not required for any activity.

   For example, if a command on a toolbar can only be used with a mouse, make sure that the equivalent command is also available through a menu or through a documented keyboard shortcut.

   The product documentation and user interface must make clear how to carry out a task using the keyboard or it must be readily inferred using knowledge of standard Windows keyboard conventions. If the documentation is not clear, the product fails the requirement to use adequately documented keyboard UI.
6. Verify that all menu items and controls have underlined access keys.

To pretest exposure of keyboard focus:

1. Use the Magnifier accessory that is included with Windows 2000. From the Start menu, choose Programs -> Accessories -> Accessibility -> Magnifier.
2. Navigate through your application using the keyboard, making sure that Magnifier accurately tracks and displays the area with the keyboard focus. In particular, verify that Magnifier:

   - Shows the correct focus location when you navigate or type within a document, text field, or table.
   - Shows the correct focus location when you navigate in menus, dialog boxes, toolbars, list boxes, or other controls.
   - Shows the correct focus location when you extend a selection in text or in a multiple-selection list box, list view, or tree view.
   - Shows the correct focus location when you return to your application from a dialog box, menu, or another application.

Chapter 4

Active Directory

Summary of Active Directory Requirements

Rationale

Many customers will deploy Active Directory in a geographically dispersed environment spanning WAN and LAN links. Hundreds or thousands of Active Directory servers that replicate with each other may be deployed in many of these environments to achieve availability and scalability.

These requirements help:

- Prevent inefficient usage of network resources.
- Ensure consistency among Active Directory-enabled applications.
- Improve supportability of Active Directory-enabled applications.
- Ensure peaceful co-existence of applications that extend the schema.

Customer Benefits

As network computing evolves to become more dynamic, static binding models become increasingly restrictive. Active Directory provides a globally available infrastructure for finding binding information by key attributes of services.

Integrating your application with Active Directory offers your customers:

- *Easier Management.* By utilizing Active Directory, you can eliminate the need for administrators to update information on the client machine whenever services are added or moved in the network.
- *Robust distributed management framework*. Since Active Directory supports multi-master replication, you can enable distributed management of your application by storing configuration information in Active Directory with built-in redundancy.
- *Granular delegated administration.* The attribute level access control in Active Directory allows your customers to delegate administration of different parts of your application to the appropriate people.

Requirements

1. Use Active Directory appropriately
2. Document the storage and replication impact of your application
3. Document your application's use of objects and attributes in the base schema
4. Information stored in Active Directory must be "globally interesting"
5. Attribute values must not exceed 500KB in length and the total object size must not exceed 1MB
6. If you extend the schema, adhere to Active Directory Schema extensibility rules

References

The Active Directory Programmer's Guide:
http://msdn.microsoft.com/developer/windows2000/adsi/actdirguide.asp

Information about Objects in Active Directory:

http://msdn.microsoft.com/library/backgrnd/html/msdn_actdsum.htm

SchemaDoc.exe utility for documenting Active Directory schema extensions
http://msdn.microsoft.com/certification/download.asp

How to Comply with Active Directory Requirements

1. Use Active Directory appropriately

This requirement helps:

- Eliminate the need for administrators to manually update applications containing client components when administrators change the network location of server components.
- Ensure that applications consume the most up-to-date information regarding network resources.
- Minimize the need to enter and maintain duplicated information about network elements.

A. If your application is comprised of multiple components that run on different machines, and there are 'client components' that request services from 'server components'[2] then:

- If Active Directory is available in the environment where your application is deployed, the server components of your application must store information in Active Directory so the client components can locate and connect to the server. This information can be a server name, RPC end point, IP address and port number or a combination of these. For client/server applications where the client and server can be developed by different vendors, it is important for the server vendor to publish information about how clients can find binding information in Active Directory.
- Applications must not store binding information in such a way that updating the network location of server components requires administrators to update each client component individually. For example, applications must not keep authoritative binding information in text files on client machines.

Examples:

- Client/Server applications where there are client components on many desktops that connect to a database server to read & write data.
- '3-Tiered' applications where there are client components on many desktops that connect to middle-tier servers that make Remote Procedure Calls (RPC) or other remote requests

[2] In this context, "client" refers to the portion of your application requesting services, and "server" is the portion of your application that provides these services. Note that is possible that in some cases, the "client" could be running on a computer running Windows 2000 Server, and a "server" could be running on a computer running Windows 2000 Professional.

- Client/Server applications where server components on workstations publish their availability to a client component on a server for periodic actions such as virus checking or file backup.
- A mail server that extends the user object with an attribute that is used to store the DNS name of the e-mail server the user is hosted on. The e-mail client uses this attribute to connect to the e-mail server hosting the user.

The following table lists examples of typical resources that can be stored in Active Directory.

Resouce	Binding Information	Example
File Service	UNC path for a share.	\\MyServer\MyShare
Server	DNS name	Myserver.example.com
Web Service	URL	http://www.microsoft.com
RPC Service	RPC binding: special encoded information used to connect to the RPC server. RPC bindings can be converted to and from strings with the RPC APIs	ncacn_ip_tcp:server.microsoft.com

Please see the following sections of Active Directory Programmer's Guide for more information on how to use Active Directory to locate network resources required by your application:

Service Publication – How to publish resources in Active Directory

Searching Active Directory – How to find resources published in Active Directory

Extending the Schema – How to extend Active directory schema to store application specific information

B. If the customer maintains authoritative information in Active Directory and your application uses this type of information, your application must be capable of using this information from Active Directory.
This applies to applications that utilize information about network elements such as users, machines, or devices. In these cases, your application must be capable of extracting this information from Active Directory. Your application must either:

- Obtain this information directly from Active Directory, OR
- If your application maintains a duplicate copy of this information, then it must provide a means to synchronize its local copy with what is in Active Directory. Specifically, your application must either:
  - Offer a synchronization mechanism within your application, OR
  - Expose APIs (or similar mechanism) that enables an external meta directory or directory synchronization product to keep the information in your application consistent with what is in Active Directory. You must document this mechanism.

Example:

- Applications that print reports containing information about users should enumerate the list of users by querying Active Directory. Applications should not keep duplicate lists of active Windows 2000 users in a

database where administrators are required to manually keep the lists up-to-date.

2. Document the storage and replication impact of your application

To provide customers with the necessary information to evaluate how your application will impact their network environment, you must document the following:

A. The incremental disk storage that your application requires for its directory objects on full replicas.
B. The incremental disk storage that your application requires for its directory objects on partial replicas (global catalog servers.)
C. The network traffic that your application requires to replicate its updates to full replicas.
D. The network traffic that your application requires to replicate its updates to partial replicas (global catalog servers.)

Specific details below.

A. The incremental disk storage that your application requires for its directory objects on full replicas.

Consider a deployment of your application on a directory that consists of a single domain running on a single domain controller. Quantify the incremental disk storage that your application requires for its directory data, i.e. the expected increase in space used in the NTDS.DIT file. If this increase is related to some parameters (e.g. the number of User objects or the selection of certain deployment options in your application) then give the storage as a function of those parameters.

If your application stores data in the configuration container then estimate configuration container usage separately from domain usage.

The formula should give an *upper bound* on the storage used, and as such need not be precise. If your application never uses more than, say, a megabyte of storage, you can say that.

In making your estimate, assume the default tombstone lifetime of 60 days.

When performing experiments to provide this estimate, use the esentutl utility to measure the space used in the NTDS.DIT file. Do not rely on the size of the file itself, since the file size includes unallocated space within the file.

To use esentutl, boot into Directory Services restore mode, open a command window, and type

```
C:> esentutl /ms /8 c:\winnt\ntds\ntds.dit
```

(substituting the actual path name of your DIT file into the line above.) The “owned” pages listed in the output are the ones in use. Each page is 8 k bytes.

The results will be more consistent/meaningful if you perform an offline defragmentation (in Directory Services restore mode, run ntdsutil, select "files" mode, run the "compact to" command) first, then run esentutl on the file that results from the defragmentation.

B. The incremental disk storage that your application requires for its directory objects on partial replicas (global catalog servers.)

Consider a deployment of your application on a directory that consists of a

two domains A and B running on two domain controllers. Suppose your application stores all its data on the first domain controller, for domain A. This data then replicates to the second domain controller, for domain B, which is a global catalog server and therefore holds a partial replica of domain A.

Estimate the incremental disk storage that your application requires for its directory data on the second domain controller, as in the previous item.

If your application stores data in the configuration container then subtract out its configuration container usage; the estimate is of the increased size of the partial replica of domain A on the global catalog server.

C. The network traffic that your application requires to replicate its updates to full replicas.

Consider a deployment of your application on a directory that consists of a single domain running on two domain controllers in one site. Suppose that your application performs all its updates against one domain controller, and Active Directory replicates those changes to the other domain controller.

Quantify the network traffic that your application requires to replicate its updates. If this traffic is related to some parameters (e.g. the number of User objects or the number of times certain application-specific operations are invoked by users), then give the traffic as a function of those parameters.

If your application stores data in the configuration container, then estimate configuration container replication separately from domain replication.

The formula should give an *upper bound* on the network traffic, and as such, need not be precise. If your application never uses more than, say, 100 kilobytes of network traffic per day, regardless of how your application is used, you can just say that.

If the network traffic generated at the time your application is installed is significantly different than the network traffic in steady-state operation of your app, then estimate the two conditions separately.

To perform this measurement, use the “DRA outbound bytes total since boot” performance counter under the “NTDS performance object”. Since this counter is a total, you want to report the difference in the values its holds before and after replication.

D. The network traffic that your application requires to replicate its updates to partial replicas (global catalog servers.)

Consider a deployment of your application on a directory that consists of two domains A and B running on two domain controllers in one site. Suppose that your application performs all its updates against the first domain controller, for domain A, and Active Directory replicates a subset of those changes to the second domain controller, for domain B, which is a global catalog server and therefore holds a partial replica of domain A.

Estimate the network traffic as in the previous item.

If your application stores data in the configuration container, then do not include configuration container replication traffic; the estimate is of the replication traffic to maintain the partial replica of domain A on the global catalog server.

Note: If your application only writes a small fixed set of objects (under 100 K bytes of data total) to the directory and only updates these objects under unusual circumstances (e.g. only when a service is moved from one machine to another), then you can just say so without providing more detailed documentation.

3. Document your application's use of objects and attributes in the base schema

Directory-enabled applications often leverage information already present in the directory; sometimes they extend this information. Customers want to understand these dependencies to help anticipate and avoid bad interactions between applications.

If your application depends upon directory objects that it does not create, you must document these dependencies:

- What optional attributes does your application read from objects that it does not create?
- What attributes does your application write to objects that it does not create?

If your application creates all of its own directory objects, this documentation requirement does not apply. However, if you create or modify anything in the configuration container, then you must fully document your usage. In this case you must specify object class, reason for the modification and/or addition, and location within the configuration container.

4. Information stored in Active Directory must be "globally interesting"

Information in Active Directory must be "globally interesting," meaning the information is useful to a large number of users in the enterprise and needs to be highly available to those users. Examples include service connection points for enterprise resources such as printers and "white pages" information such as telephone numbers, pager numbers and e-mail addresses, so users in the enterprise can find the information easily and efficiently.

By contrast, Active Directory is not an appropriate store for information that is of local interest only, since replication of Active Directory consumes network bandwidth. For example, items that should not be stored in Active Directory include the names of local files on the user's computer, the user's currently selected e-mail message, and perfmon data.

5. Attribute values must not exceed 500KB in length and the total object size must not exceed 1MB

Active Directory does not support streaming of attribute values and is not appropriate for very large data. For this reason the maximum length of an attribute value stored in Active Directory must not exceed 500KB in size and the total object size must not exceed 1MB.

6. If you extend the schema, adhere to the schema extensibility rules

The purpose of this requirement is to ensure that applications that extend the schema do so in a consistent manner and that schema extensions are well documented. Schema extensions can be a potential source of conflict among applications because schema extensions are designed independently by different software vendors and application writers. Microsoft recommends that customers strictly control schema updates in their organization.

If your application extends the Active Directory schema, it must adhere to the rules specified in this section to prevent such conflicts. The rules are divided into

the following categories:

A. Installation of schema extension

B. Documentation of the schema

C. Specification of the Schema-ID-GUID

D. Schema Object Naming

E. Specification of Link ID

Please see the “Extending the Schema” chapter of Active Directory Programmer’s Guide for more information on schema extensibility.

A. Installation of Schema Extensions

If your application requires schema extensions:

- The installation of the schema extension must be performed either by a separate program or with a separate option in your regular installation program.
- The schema installation program must allow the user to exit the program prior to making any changes to the schema.

This allows customers to perform the schema update separately and in advance of the rest of the installation.

We recommend that schema extensions be made programmatically in most cases, though other methods, such as use of LDIF scripts, may be used.

B. Documentation of the schema

You must provide documentation of your schema extensions in the form of a well-formed and valid extensible markup language (XML) document[3]. This documentation must be available on your Web site. A pointer to this Web site must be present in the product documentation that accompanies the application.

You must use the utility schemadoc.exe to generate the XML-formatted documentation for your schema extensions. This tool is available at http://msdn.microsoft.com/certification/download.asp. Given the DNS name used to name your schema elements, this utility will generate a well-formed XML file with information about your schema extensions and a tag representing each of the three pieces of information that you need to provide.

- Description of each new class and attribute with its expected use.
- Expected update frequency and expected size of the attribute, which allows the customer to make calculations of replication impact.
- Envisioned security privileges required to administer (create instances, make changes to values, delete instances) the attribute/class.

Note: To help customers identify the impact of installing an application in their environment, reduce the time to deployment, and reduce support calls to you, Microsoft will provide a directory of schema extensions made by Windows 2000 Certified applications. This directory will contain links to your schema documentation on your website.

C. Specification of the Schema-ID-GUID

[3] An XML "document" is a set of XML tags and content constructed according to the rules of XML. For Certification, the document must be *well formed*, meaning that it conforms to the rules of XML, and must also be valid, meaning that it conforms to a *schema*. See http://www.w3.org/ for the latest information on XML standards.

For efficiency reasons, each Active Directory schema object is identified by a Globally Unique Identifier (GUID) in addition to an OID. GUIDs are fixed length and can be dealt with more efficiently than OIDs. Active Directory uses OIDs for interoperability with external clients and GUIDs internally for efficiency.

When extending the schema, you must specify the value of the schemaIDGUID attribute on each classSchema and attributeSchema object in addition to the OID. This ensures the schemaIDGUID attribute is identical across Active Directory installations.

D. Schema Object Naming

You must adhere to the following rules with respect to schema naming for both the common-name (cn) and the LDAP display name (ldapDisplayName) of schema objects.

- Start with the name of your company in lower case.
- The next token in the CN must be a hyphen.
- Following the hyphen, specify an attribute or class name which is unique within your company.

The following table illustrates an example:

Company Name	Attribute or Class Name	Common-Name (cn)	LDAP-Display-Name (lDAPDisplayName)
Microsoft	VoiceMailID	microsoft-VoiceMailID	microsoft-VoiceMailID

You must register this name with Microsoft, using the following Web site: http://msdn.microsoft.com/certification/ad-registration.asp. There will be a nominal charge for registration to cover costs and your registration is only complete after you receive notification from Microsoft that it is approved.

Exception:

If you have existing proprietary LDAP names already established for your application or its predecessor, your application may be exempted from this naming convention, provided that you register the existing proprietary names with Microsoft by June 1st, 2000. To apply for this exception, see http://msdn.microsoft.com/certification/ad-registration.asp.

Note, the documentation requirements above still apply.

E. Specification of Link-ID

Active Directory supports attribute values which reference other objects in Active Directory. This provides a way of representing relationships among objects in Active Directory. For example, members of a group are linked to the objects that represent the members, or the manager attribute of a user is linked to the object representing the manager of the user. Using linked attributes, you can find the groups that a user is a member of or the direct reports of a manager.

If you require linked attributes, you must obtain a Link-ID from Microsoft using the tool at http://msdn.microsoft.com/certification/ad-registration.asp.

How to Pretest Applications for Active Directory Support

To pre-test appropriate usage of Active Directory (Requirement 1A)

- Install Active Directory, by running DCPROMO on a Windows 2000 server
- Install your application
- Verify that the client side of your application binds to the server resource
- Modify the server binding information in Active directory so the data is different, but remains valid
- Verify that the client side of your application binds to the "new" server resource and not a cached copy of the "old" resource
- Modify the server binding information in Active directory so some data is invalid
- Verify that the client side of your application does not bind to the server resource

To pre-test that your application is capable of using information from Active Directory (Requirement 1B)

If you are duplicating information that is in Active Directory and Active Directory is the authoritative source for this information:

- Ensure that you can synchronize data that is duplicated in Active Directory with information in your application's data store OR
- Verify that you have a documented a way of getting at this information from your application data store

To pre-test that your application adheres to schema extensibility rules (Requirement 6)

A. Use the schema extension option of your setup program to install your application's schema extensions. Then on a different machine install the application (this time without running the schema extension option).
B. Use schemadoc.exe to export your schema extensions out of Active Directory and ensure it is the same as that on your web site and that it is complete. Also verify that the documentation that accompanies your application includes a link to this web site.
C. Install your application on a different Active Directory forest and ensure the schema-ID-GUID for each schema extension is the same for the two forests. You can do this by comparing the schema document generated by running schemadoc.exe on each of the forests.
E. Confirm that the link IDs you have used are those you got from Microsoft.

Chapter 5

Security Services

Summary of Security Requirements

Rationale

In order for Client/Sever applications to provide secure access to resources and security for their interactions, the application must configure and run the client and server portions in the correct security contexts. This provides access to the resources needed in a secure manner and provides a secure communication channel between the client and the server. The proper configuration depends on:

- Running the server in the appropriate security context.
- Providing connection authentication between the client and the server.
- Using standard security credentials and security descriptors to validate access to resources and data.

If a service or service account password is compromised, then the scope of damage that can occur is limited to the capabilities of the account under which the service is running. To minimize such damage, service accounts should run with the least privileges necessary to perform the functions required by the service. Running the server in the appropriate security context also allows the administrator to easily provide desired partitioning of access to resources.

Single Sign-On (SSO) allows enterprise network users to seamlessly access all authorized network resources, on the basis of a single authentication that is performed when they initially access the network. SSO can improve the productivity of network users, reduce the cost of network operations, and improve network security.

Customer Benefits

Documenting the services that require more than User-level privileges provides customers with the information necessary to make one of the following choices:

- Run the service under a non-Admin context and grant explicit permissions to that service account in order to allow specific features to succeed.
- Run the service under an Admin (or Domain Admin) context and accept the inherent security risks.
- Deploy a different product that offers similar functionality that can run under a more secure context.

Single Sign-On (SSO) allows users to access all authorized network resources on the basis of a single authentication that is performed when they initially access the network. This results in the following benefits:

- **Better network security.** All SSO methods available under Windows 2000 provide secure authentication and provide a basis for encrypting the user's session with the network resource. Eliminating multiple passwords also reduces a common source of security breaches—users writing down their

passwords.

- **Improved user productivity.** Users are no longer required to remember multiple logons, nor are they required to remember multiple passwords in order to access network resources. This is also a benefit to help desk personnel, who need to field fewer requests for forgotten passwords.
- **Simpler administration.** SSO-related tasks are performed transparently as part of normal maintenance, using the same tools that are used for other administrative tasks.
- **Better administrative control.** All SSO-specific information is stored in a single repository, the Active Directory. Because there is a single, authoritative listing of each user's rights and privileges, the administrator can change a user's privileges and know that the results will propagate network wide.
- **Consolidation of heterogeneous networks.** By joining disparate networks, administrative efforts can be consolidated, ensuring that administrative best practices and corporate security policies are being consistently enforced.

Requirements

1. Document services that require more than User level privileges to run
2. Win32 clients running in the context of a trusted domain account must support Single Sign-On

References

Single Sign-On in Windows 2000 Networks:
http://www.microsoft.com/security/Resources/SSOWhite.asp

Additional Windows 2000 Security Whitepapers
http://www.microsoft.com/windows/server/Technical/security/

How to Comply with Security Requirements

1. Document services that require more than User level privileges to run

Services execute in the context of a "service account". If a service or service account password is compromised, then the scope of damage that can occur is proportional to the capabilities of the account under which the service is running. Therefore, service accounts should run with the least privileges necessary to perform the functions required by the service.

For each service account that requires more than User-level privileges, the following must be documented:

1. The specific system resources that normal Users do not have sufficient access to by default.
2. The specific User Rights that normal Users do not have by default in order for the service to run properly.
3. The minimum permissions to the resources identified in #1 that the service account must have in order to run properly.

For example:

Service	Resource or User Right	Minimum resource permission required by service
Fax	%systemroot%	Write (This Directory, subdirectories and files)
Mail	HKLM\Software\CompanyA\Mail	Write (This directory, subdirectories and files)
Backup	Backup and Restore Privilege	N/A
Software Distribution	Admin Shares	Read\Write
Web Server	HTTP Service	Stop\Start
Disk Manager	NTFS Volumes	Open Handle for Modify

Note that some applications may unjustifiably require elevated privileges for a service. Consider the first three hypothetical examples in the above table.

Fax Service - Assume that the Fax Service is using the %systemroot% directory to store incoming faxes. While it is true that administrative level privileges are needed to write to %systemroot%, there is no justifiable reason for the fax service to use this directory as a storage location in the first place. If the faxes were stored in a non-system location, the service may not require administrative level privileges in order to run.

Mail - Assume that the Mail service only needs to retrieve some configuration parameters from the `HKLM\Software\CompanyA\Mail` key. In this case, it does not need to open the key for write access and therefore requiring write access to this key is unjustified.

Backup – Assume that the Backup Service runs under an administrative context because administrators have the backup and restore privilege by default. This is not justified because the Backup and Restore privilege could be explicitly granted to a non-admin account. The Backup Service could then run under the non-admin account and accomplish its tasks without inheriting all of the other permissions that are given to administrators by default.

NOTE:

The Certification program does not require any specific default service account setting, nor does it evaluate the appropriateness of using elevated privileges. You must document your need for elevated privileges to provide customers with the information necessary to make one of the following choices:

- Run the service under a non-Admin context and grant explicit permissions to that service account in order to allow specific features to succeed.
- Run the service under an Admin (or Domain Admin) context and accept the inherent security risks.
- Deploy a different product that offers similar functionality that can run under a more secure context.

2. Win32 clients running in the context of a trusted domain account must support Single Sign-On

Win32 clients must correctly support Single Sign-On, which means that when a user is logged on in the context of an account in a trusted domain:

- The client does not prompt for logon credentials.
- The client uses the credentials the user logged on with.
- The Windows 2000 server application authenticates the client as that user.

Note: It is acceptable (and sometimes desirable) to support logon with alternate credentials as needed, provided the client also supports Single Sign-On. For example, the "net use" command supports a “/u:” option to allow use of alternate credentials when authenticating to a remote file server. Additionally, the Windows shell supports a "Run As..." option to start a process in a different logon context.

Implementation Details

You can implement Single Sign-On by using any Windows networking protocols that support integrated authentication, such as RPC, DCOM, and named pipes.

- If you are using named pipes, you will automatically pass this requirement.
- Using RPC calls (**RpcBindingSetAuthInfo** and **RpcServerRegisterAuthInfo**), you can set a particular authentication provider. SSPI supports multiple providers - NTLM, Kerberos, SSL, etc. Any one of these providers can be used for single-sign on. We recommend you use `RPC_C_AUTHN_GSS_NEGOTIATE` (for automatically negotiating Kerberos or NTLM) or `RPC_C_AUTHN_GSS_SCHANNEL` (for Public key authentication).
- DCOM provides options similar to RPC (**CoInitializeSecurity**). See Windows SDK for details.

If you do not use any of these protocols, then you should use the Security Support Provider Interface (SSPI) on top of your application protocol. See the Security section in the Windows SDK for details.

Best Practice

While not required for Certification, it is recommended that the server impersonate the client’s security context to access objects and resources outside the server’s immediate control (such as other local files and the registry) when performing operations on behalf of the client. This ensures that the proper security context is used to validate the operation. However, if the server implements its own access control model for server-private data, then client impersonation is not needed, but the authenticated user identity should be used to determine access permissions. It is recommended that server applications leverage the Windows 2000 access control model by storing permissions in security descriptors using Win32 Private Object Security APIs and checking them using Win32 Access Check APIs.

Client impersonation by the server is implemented using the access token in the negotiated security context. In order to accomplish this, the server calls **ImpersonateSecurityContext** passing the handle of the client security context. When this call returns successfully, the server thread that made the call has assumed the identity of the client. Any access to system resources or data or to data or resources maintained by the server can be validated against standard Windows 2000 Security Descriptors.

When the server wishes to return to using its own security context, then it should call **RevertSecurityContext**. For example:

```
// Server impersonates the client using the security context that was
// created during the mutual authentication.
ImpersonateSecurityContext (pContextHandle);

// Sever gets the Windows 2000 standard security descriptor for its
// private object using a server routine. The object is stored and
// manipulated by server-specific routines, but the Security Descriptor
// associated with the data is the Windows 2000 standard format.
MyStatus = GetObjectSD(Object,…,&SD);

// Obtain client's token. Since we are impersonating the client, the token
// returned is the client access token.
Status = OpenThreadToken(…,&Token);

// Perform Access Check using the Windows 2000 standard access check API.
// Since the SD on the private object is Windows 200 standard Security
// Descriptor, the Windows API can make the access check.
Status = AccessCheck( SD, Token, DesiredAccess, GenericMapping,
            &PrivsUsed, &PrivLength, &GrantedAccess,
            &Allowed);
// Act as per the result
if (Allowed) {
}
```

How to Pretest Applications for Security Support

To test if your server runs in the correct user context

1. Clean-Install Windows 2000 Server onto an NTFS partition joining a domain as necessary
2. Log in as an Administrator
3. Install the application
4. If the service account created/used by setup is
   - NOT the local system account, and
   - NOT a member of the Administrators group, and
   - NOT a member of the Domain Administrators group, and
   - NOT a member of the Power Users group

   Then,

   4a. Verify that any User Rights granted to the service account are properly documented.

   4b. Quit.

   Otherwise, continue on with step 5.
5. Create local or domain user accounts for each service that will be installed. Create local accounts if the service does not need network access, otherwise create domain accounts

6. If necessary, reconfigure the service(s) to run as a member of the local User's group
    a. If a service is configured to run as local system, use the services applet to run the service[4] under an account created in step 5.
    b. If a service is configured to run under the local administrator account, use the services applet to run the service under an account created in step 5.
7. Remove all members of the local Administrators group except for Administrator and Domain Administrators
8. Remove all members of the Domain Administrators group except for the Administrator.
9. Remove all members of the Power Users group.
10. As specified in the your documentation, explicitly grant all User rights required by the service to the user account that the service is running under. If the setup program has already accomplished this, then verify that the user rights granted by setup are the same as the user rights listed in the ISV documentation. If not, the documentation must be updated.
11. As specified in the your documentation, explicitly grant all permissions to all resources required by the service accounts on the both local and remote machines.

    Note: It may not be possible to completely perform this task. If a service account requires permissions that are "hardwired" only for the Administrators group, then there will be no way to grant those permissions to the service account. Such permissions include:

    - The ability to access admin only ($) shares.
    - The ability to install other services or drivers.
    - The ability to read/write device drivers via IOCTLS
12. Verify that the application functions properly.
13. If the application requires "hardwired" administrative privileges such as those listed in the note to step 11, verify that the service fails when those permissions are requested.

How to pre-test support for Single Sign-On

If the server-side of the application has an option for anonymous or authenticated logon, select authenticated logon. If the server does not have this option, use the default.

- Run the client in the context of an anonymous, unauthenticated account and try to access the server. The client should either prompt you for an authenticated user logon or deny you access to resources.
- Run the client in the context of an authenticated trusted domain account. The client should provide access to expected resources as per the access control policy, without prompting you for a logon.

[4] To run a service under a specific account, 1) right click on My Computer and select "Manage" 2) Expand the "Services and Applications" node. 3) Select the "Services" node in the left pane. 4) In the right pane, double click the service for which you want to adjust the account properties. 5) Select the "Log On" tab. Select the "This Account" radio button. Type in the account you wish to use.

Note: If the server allows anonymous, unauthenticated logon by default and there's no option to allow only authenticated logon, the application cannot be Certified for Windows 2000.

Chapter 6

Cluster Service

Summary of Cluster Service Requirements

NOTE: This chapter is required for Certification on Windows 2000 Advanced Server and Windows 2000 Datacenter Server [5]. Applications that do not meet these requirements are eligible for the Windows 2000 Server certification only.

Rationale

A server cluster is a group of independent servers managed as a single system for higher availability. Cluster Service is a set of system services in Windows 2000 Advanced Server and Windows 2000 Datacenter Server that enables you to form server clusters by connecting multiple servers together, making them appear to network clients as a single, highly available system.

Cluster Service can automatically detect the failure of an application or server, and restart the application, either on the same server if it is still alive, or on another surviving server.

These requirements help ensure that your application will run properly with Cluster Service enabled, so that:

- Your server application can failover to other servers
- The client-side of your application properly handles failure of the server application

Customer Benefits

Customers that run your application in a clustered environment can achieve higher availability, because your application can continue to provide service during both planned downtime (such as hardware and software upgrades) and unplanned outages (such as hardware or software failure).

When one of the systems—or nodes—in the cluster fails or becomes unavailable, Cluster Service transfers its workload to another system in the cluster. Users only experience a momentary pause in service. Cluster Service can also be configured to provide failback, so that when the failed server comes back online, the workload is rebalanced across the server cluster.

Requirements

1. Applications must be able to install on at least 2 nodes for certification on Windows 2000 Advanced Server and 2,3, and 4 nodes on Windows 2000 Datacenter Server.
2. Application must support failover to all cluster members

[5] Windows 2000 Datacenter Server is on a different release schedule than the other Windows 2000 operating systems. Microsoft is currently evaluating whether any additional requirements specific to Datacenter will be required for Certification on Windows 2000 Datacenter server.

3. Clients must survive failure of the server application without crashing or affecting the stability of the system

References

Cluster Service Architecture Whitepaper
http://www.microsoft.com/windows/server/Technical/management/default.asp

Clustering in Windows 2000 Advanced Server:
http://www.microsoft.com/ntserver/ntserverenterprise/exec/overview/Clustering/ASOverview.asp

White papers on Windows 2000 Clustering:
http://www.microsoft.com/ntserver/ntserverenterprise/exec/overview/clustering/default.asp

Information on writing resource DLLs:
http://www.microsoft.com/ntserver/ntserverenterprise/techdetails/moreinfo/ClustAwareApps.asp

"About Server Clusters" in the Platform SDK:
http://msdn.microsoft.com/isapi/msdnlib.idc?theURL=/library/psdk/mscs/cs_about_2var.htm

General Background:
In Search of Clusters by Gregory F. Pfister, ISBN: 0-13-437625-0. An excellent introduction to clustering technology including a description of the common programming models.

How to Comply with Cluster Service Requirements

1. Applications must be able to install on 2 nodes for certification on Windows 2000 Advanced Server and 2,3, and 4 nodes on Windows 2000 Datacenter Server.

In order to qualify for Certification on Windows 2000 Advanced Server, applications must install on 2 nodes.

In order to qualify for Certification on Windows 2000 Datacenter Server, applications must install on 2 nodes, 3 nodes, and 4 nodes.

Note: Your application setup should not make any assumptions about the number of nodes in the cluster. It should enumerate all nodes in the cluster and allow installing your application on any node, even if the disks where your application data is stored are not physically located on that node.

2. Applications must support failover to all cluster nodes

When a node in the cluster fails, Cluster Service will move the resource group from that node to a new node. A resource group is a collection of resources that provide services to clients and can depend on each other.

Resources that represent the primary functionality of your application must be able to start up (come online) on any other node in the cluster. After a failover is complete, clients should be able to access all data exposed by primary functions.

Note: Cluster Service operates under a "shared nothing" architecture in which each server owns its own disk resources. In the event of a server failure, ownership of the clustered disk is transferred from one server to another. For applications to properly support failover, the application's data must be stored on the clustered disk.

3. Clients must survive failure of the server application without crashing or affecting the stability of the system

Any client that ships with your server application must gracefully handle both cluster node failures and application failures. Cluster and application failures may cause clients to temporarily lose their connection to the server application (see below). Your client must survive both the failure of the server application and node failure as follows:

- When a connection to the server application is lost, your client application must not crash or compromise the stability of the client operating system.
- Once the failover is complete and the application is restarted on a cluster node, your client must reconnect to the cluster using either of the following mechanisms:
  1. Re-establish the lost connection without user intervention and with no loss of data, OR
  2. Offer the user a chance to reconnect and retry the operation that failed - for example, prompt the user to refresh the data in the client.
- If the server application is not able to restart, the client must inform the user that the connection could not be re-established.

Connections to the server application can be lost for any of the following reasons:

- Application fails and is then restarted on the same node
- Application fails and is restarted on a new node
- Node fails, and all resources failover to a new node
- Administrator moves the resource group containing the application to a new node
- The administrator shuts down the server application
- All nodes in the cluster fail
- The client's network connection to the cluster is interrupted, even though the cluster and the server application are still running

These failures may be exposed to the client application as application timeouts, invalid handles, network failures and connection timeouts.

Development Guidelines:

The guidelines in this section are not requirements that will be tested individually for Certification. However, following these guidelines will help you meet the requirements described above.

1. Use TCP/IP protocol

Services that communicate with clients (as well as their clients) must use TCP/IP in order to be able to take advantage of IP address failover provided by Cluster Service. Servers that do not communicate with clients need not use TCP/IP.

2. Application should use a virtual server name and IP address to connect to the node hosting the server application.

Clients communicating with the cluster resources must use the virtual server IP address or virtual server network name to support failover.

If the server application publishes a network name or IP address to clients, it must publish IP virtual server IP address or network name. A server application that depends on a computer name or an IP address should use a

network name and/or an IP address of a virtual server that is used by clients to access this application. Your server application should not fail to restart on another node because the computer name on this node is different.

The following code sample illustrates how to set the server application environment as part of your resource dll online routine.

```
//
// Create the new environment with the simulated net name when the
// services queries GetComputerName.
//
if ( ! ClusWorkerCheckTerminate( pWorker ) )
{
   nStatus = ResUtilSetResourceServiceEnvironment(
      YOUR_SERVICE_NAME,
      pResourceEntry->hResource,
      g_pfnLogEvent,
      pResourceEntry->hResourceHandle
      );
   if ( nStatus != ERROR_SUCCESS )
   {
      break;
   } // if: error setting the environment for the service
}
```

About IP address failover:

Client applications use a *virtual server* IP address to access services running on a Windows 2000 Server Cluster. A Virtual Server is a cluster resource group containing an IP address and a network name. A virtual server can be brought online on any node in the cluster, however, it appears to clients accessing it as the same physical machine.

The IP address of the *virtual server* has to be configured as a cluster resource in the same resource group where the server application was created. In case of a node failure, all resource groups running on this node are moved to another node in the cluster. The IP address of the virtual server is now available on another node and all connections with the clients can be reestablished.

3. Upon failure, clients must preserve user data

The client application must be able to reconnect and resume an operation in the event of cluster node failure or application. It must either offer the user a chance to retry the connection or it must retry the connection automatically until it succeeds or can determine that the server application could not be brought online.

In case of a node failure, all resource groups running on the failed node are moved to another node in the cluster. Cluster Service requires some time to bring all resources online and restart services on the other node. The time needed to fail over a server application depends on many factors. The most significant is the time required to restart the application.

4. Location of application data must be configurable

Cluster service can fail over only disks managed by the cluster that are on the storage bus shared among all nodes in the cluster. Your application setup should allow selecting the drive and installing application data on any drive. Cluster aware setup should allow installing the application data only on a shared drive managed by the cluster.

5. Checkpoint either automatically or manually state information required

for clean restart

If a server application maintains any state information required for a clean restart, it should checkpoint this state information frequently to a shared disk managed by a cluster. It should use this data to quickly recover after a failure.

6. Upon failure application can be restarted and, if applicable, recover to the last checkpoint

The server application must recover from a node failure. A sudden node failure, for example a power blackout, should not leave your application in a state where it cannot restart.

After a node failure, Cluster Service moves the server application running on the node to another node along with any other resources it may depend on. The server application must restart, recover, and resume operation in time you specified in your product literature.

7. At least one instance of the application can run as a cluster resource

Cluster Service manages applications as cluster resources. A cluster resource is a physical or logical entity that can be owned by a node, brought online and taken offline, moved between nodes, and managed as a server cluster object. A resource can only be owned by a single node at any point in time. A resource is associated with, and managed by, a resource type.

If the resource supports it, Cluster Service can manage multiple instances of the same resource, but it is acceptable to support only one instance.

To take advantage of clustering, your application has to be configured as a cluster resource. You should be able to create at least one instance of your application. Your application must function properly as a cluster resource. Cluster service must be used to start (bring online) and stop (take offline) your application.

8. Can be configured at least as a generic service or application

The monitoring and failover capabilities of Cluster Service can be extended to support any application. Cluster Service uses resource DLLs to extend its failover support to other resource types.

Applications that do not offer an application specific resource DLL can still take advantage of clustering by using a generic application or generic service resource type. These resource types offer failover protection against most failures, notably, node failure. However, they cannot detect your application failures. If your application crashes or hangs, Cluster Service won't be able to detect this failure and either restart or failover your application.

It is acceptable to use generic application or generic service type to manage your application as a cluster resource.

How to Pretest Applications for Cluster Service Requirements

How to pretest that your application is cluster ready.

If your application's setup is cluster aware, use setup to configure all nodes.

If your application's setup is not cluster aware, install your server application on at least 2 nodes in the cluster. Use the Cluster Administrator console to create a virtual server and configure your server application as a generic service or application. Use Cluster Administrator console to move your resource to either node in the cluster. If your application is cluster ready it should come online on

any node in the cluster. Clients should be able to access the service provided by your application, no matter which node hosts it.

For certification on Datacenter, repeat this procedure for 3 node and 4 node configurations.

How to pretest that your application supports failover

1. Once you have installed the application on all nodes in the cluster, run functionality tests to verify application is fully functional and stable.
2. Cause the node running your application to fail so that fail-over of the application is triggered. Below are suggested techniques to trigger failure:

   a. HW failure - simulate by doing hard reset

   b. OS failure - simulate by emitting a Ctrl+C command followed by a ".reboot" command from a remote kernel debugger.

   c. Application failure - simulate using the End Process feature in Task Manager or Process Viewer (Pview.exe in the Windows SDK).

   Note that normal shutdown of the machine is not a valid test for failover, since the application will have the opportunity to gracefully shutdown.
3. Verify that application restarts on a new node in the cluster.
4. Run functionality tests to verify all functionality is again available on the new node. The application must have access to all data that it previously had access to.
5. For testing on Datacenter, repeat steps 2-4 to verify that the application subsequently fails over to each of the remaining nodes.

How to pretest that clients you provide survive failure and subsequent restart of the server application

Cause the server application to fail using each of the following scenarios.

1. Shutdown the server application using the normal shutdown sequence and leave all nodes in the cluster running.
2. Terminate the application process (do not use the normal shutdown sequence), but leave the node running
3. Kill the node. Note: Do not use normal shutdown. The node and application must not have time to exit gracefully. Below are suggested techniques for various failure modes:

   a. HW failure - simulate by doing hard reset

   b. OS failure - simulate by emitting a Ctrl+C command followed by a ".reboot" command from a remote kernel debugger

   c. Application failure - simulate using the End Process feature in Task Manager or Process Viewer (Pview.exe in the Windows SDK).

For each case,

- Verify that the client does not crash or lose stability when the server application fails.
- Once the server application is restarted, either on the same node or a different node, verify that the client either:
  - Re-establishes the connection with no user intervention and no loss of user data, OR

- Prompts the user to retry the connection and that the application then establishes the connection.

Note: if you need to manually configure the client to access the server application, configure it to use the virtual server, not the node name.

How to pretest that clients you provide survive failure without subsequent restart of the server application

1. Cause the application to fail in a way that does not allow it to restart on the cluster. To do this, you can take the resource offline, or kill both nodes.
2. Verify that the clients do not crash or lose stability.
3. Verify that the clients notify the user in a reasonable time that the connection to the server application was lost.
4. Verify that the client can be closed without crashing or affecting the stability of the client's workstation, and that the user can preserve data, if appropriate.

Appendix A

Best Practices

Important: The guidelines outlined in this appendix are not required to comply with the Application Specification for Windows 2000, but are strongly encouraged to provide a better user experience.

A-1 Do not require a reboot for installation of your application on Windows 2000

A-2 Use SHGetFolderPath to determine special folder paths

A-3 Test your application under Terminal Services

A-4 Globalization

A-5 Localizability

A-6 Use 64-bit compatible data types

A-7 Additional Considerations for Cluster Service

A-8 Windows Management Instrumentation

A-9 Provide MMC snap-in for management tools

A-10 Expose a COM-based scripting model

A-11 Use COM+ for Distributed Applications

A-12. Run without NetBIOS in Windows 2000-only environment

A-1 Do not require a reboot for installation of your application on Windows 2000

Installation of your application should not require a reboot when you install it onto a freshly installed Windows 2000 system with no applications running unless you are deploying any of the following as part of your install.

- Operating system update package, such as service pack
- System-level drivers, such as 3rd party video drivers, disk drivers, and administrative level privileged services.

Generally, a reboot on Windows 2000 should not be necessary, even if applications are running. In the past a typical reason applications would reboot was because they were replacing a file that was in use at the time. However, on Windows 2000, these situations are greatly reduced by Windows File Protection (WFP).

WFP protects essential system files, so applications should not attempt to replace them. These files will be updated only via an operating system update package, instead of in a piecemeal fashion by individual applications.

In addition, applications that use the Windows Installer will be less likely to need a reboot. Windows Installer service will automatically check to see if other applications or service processes are using files that it is attempting to update. In these cases Windows installer will prompt the user to shutdown the applications that are using those files. If the user does this, the application can install without a reboot. If the user does not shut down those applications, Windows Installer will prompt for a reboot.

Based on strong customer feedback, we recommend that you **DO NOT REQUIRE A REBOOT UNLESS IT IS ABSOLUTELY NECESSARY.**

A-2 Use SHGetFolderPath to determine special folder paths

Whenever you access any of the special folders in the following list, your application should use the Win32 APIs to dynamically obtain the proper language-specific folder names. The preferred way to do this is using the **SHGetFolderPath** API with the appropriate CSIDL constant. This function behaves consistently across Windows 95, Windows 98, Windows NT 4.0, and Windows 2000. This API is redistributable via the SHFOLDER.DLL. Software vendors are encouraged to redistribute this component as much as possible to enable this support on Windows operating systems prior to Windows 2000. Windows 2000 includes this DLL as a protected system file and, as such, this DLL cannot be replaced on Windows 2000 or greater.

Note: To ensure your application can run on Windows 9x, Windows NT 4 and Windows 2000, always link to the SHGetFolderPath implementation in SHFOLDER.DLL. Windows 2000 natively implements SHGetFolderPath in SHELL32.DLL, but other versions of Windows do not include SHGetFolderPath in SHELL32.DLL.

Standard folder	*CSIDL constant name*
Alternate Startup ([user], DBCS)	CSIDL_ALTSTARTUP
Alternate Startup folder (All Users profile, DBCS)	CSIDL_COMMON_ALTSTARTUP
Application Data ([user] profile)	CSIDL_APPDATA
Application Data (All Users Profile)	CSIDL_COMMON_APPDATA
Control Panel virtual folder	CSIDL_CONTROLS
Cookies folder	CSIDL_COOKIES
Desktop (namespace root)	CSIDL_DESKTOP
Desktop folder ([user] profile)	CSIDL_DESKTOPDIRECTORY
Desktop folder (All Users profile)	CSIDL_COMMON_DESKTOPDIRECTORY
Favorites folder ([user] profile)	CSIDL_FAVORITES
Favorites folder (All Users profile)	CSIDL_COMMON_FAVORITES
Fonts virtual folder	CSIDL_FONTS
History folder	CSIDL_HISTORY
Internet Cache folder	CSIDL_INTERNET_CACHE
Internet virtual folder	CSIDL_INTERNET
Local (non-roaming) data repository for apps	CSIDL_LOCAL_APPDATA
My Computer virtual folder	CSIDL_DRIVES
My Pictures folder	CSIDL_MYPICTURES
Network Neighborhood directory	CSIDL_NETHOOD
Network Neighborhood root	CSIDL_NETWORK
Personal folder ([user] profile)	CSIDL_PERSONAL
Printers virtual folder	CSIDL_PRINTERS
PrintHood folder ([user] profile)	CSIDL_PRINTHOOD
Program Files folder	CSIDL_PROGRAM_FILES
Program Files folder for x86 apps on Alpha systems	CSIDL_PROGRAM_FILESX86
Programs folder (under Start menu in [user] profile)	CSIDL_PROGRAMS
Programs folder (under Start menu in All Users profile)	CSIDL_COMMON_PROGRAMS
Recent folder ([user] profile)	CSIDL_RECENT
Recycle Bin folder	CSIDL_BITBUCKET
SendTo folder ([user] profile)	CSIDL_SENDTO
Start menu ([user] profile)	CSIDL_STARTMENU
Start menu (All Users profile)	CSIDL_COMMON_STARTMENU
Startup folder ([user] profile)	CSIDL_STARTUP
Startup folder (All Users profile)	CSIDL_COMMON_STARTUP
System folder	CSIDL_SYSTEM
System folder for x86 applications on Alpha systems	CSIDL_SYSTEMx86
Templates folder ([user] profile)	CSIDL_TEMPLATES
User's profile folder	CSIDL_PROFILE
Windows directory or SYSROOT	CSIDL_WINDOWS

A-3 Test your application under Terminal Services

Many enterprise customers use Terminal Services to provide Windows 2000 applications to an array of desktop and mobile clients. Applications running under terminal services run only on the server. The terminal server client performs no local processing of applications. This means that in a terminal services environment, multiple instances of the same application may be running on the same machine at the same time, serving different users.

Considerations for developers

Because multiple users are running your application at the same time, it's critical to store application and user data properly. For more information, see Chapter 4, "Data and Settings Management" in the Desktop Application Specification.

In a Terminal Services environment, multiple instances of your application will be running on the same machine. Your application's .EXE and .DLL files should be written so multiple users can simultaneously run your application on the same machine. Considerations include:

- *File locking*: Ensure that files are not locked during use, as this could prevent multiple instances of the application, or processes under the application such as wizards, from running.
- *File permissions*: Users may not have access to system files, and may not have the same permission levels as the administrator who installed the application.
- *File locations*: Per user data and configuration files should be stored separately to avoid collisions and manage permissions. In particular, applications should store temporary information on a per user basis to avoid conflicts among users' information and preferences. You should do this by using the GetTempPath API rather than using a hard-coded path.

Terminal Services manages objects on a per-client basis for you. You only need to implement object management if you do not want the operating system to do this for you. If a specific object should be available to all instances of the application, register the .DLL or .EXE that creates the object with the command "register filename /system", or change the application to prefix the case-sensitive string Global\ (in C string form "Global\\") to the name of the object when it is created.

Pretesting Guidelines

1. Set up a Windows 2000 Server computer as a Terminal Server Host and configure it to allow both administrator and non-administrator access.
2. Install the application on this machine logged on as an Administrator.
    - The application should be installed in Terminal Server's Install mode. Use the command line "change user /install" or use Add/Remove Programs in the Control Panel. This is required so that Terminal Server can appropriately manage the registry for each user. An installation script may also be required to adjust file and registry settings for multiple users. See the whitepaper, "Using and Developing Applications Compatibility Scripts."
3. On a separate client machine, log on to the Terminal Server Host with normal (non-Power, non-Admin) User privileges.
4. Ensure that all shortcuts, files and folders are available to the user.

5. Launch each program in the application.
6. Run a full set of functionality tests on your application.
7. Create customized interface, file option and new default settings for the user.
8. Log on as a different user and make sure that the setting changes you made for the first user are not the new defaults for the new user.
9. Run a full set of functionality tests on your application simultaneously from two client computers as two different users.
10. Run a full set of functionality tests on your application simultaneously from two client computers with one (non-Power, non-Admin) User and one Administrator.

Diagnosing Common Problems

- If the problem was encountered while logged on to a client as a user without administrative rights, try to reproduce the problem on the Terminal Services server console as the same user. If the problem goes away, this indicates that the application is using a system resource or configuration. The console session has all system processes, such as SMSS.exe, RPCSS.exe, SERVICES.exe, and LSASS.exe, and the Windows 2000 services associated with it.
- If the problem is still present on the Terminal Services server console, log on to a client machine as an Administrator and repeat the test. If the error does not occur, the problem is probably a file permissions problem.
- If the problem is still present, log on to the Terminal Services server console as an Administrator and repeat the test. If the error does not occur, the problem may be caused by incorrect permissions or the application is using a system resource or configuration.

References

Optimizing Applications for Windows 2000 Terminal Services and Windows NT Server 4.0, Terminal Server Edition:

http://www.microsoft.com/windows/server/Technical/terminal/TSAppDev.asp

Using and Understanding APIs for Terminal Server:

http://www.microsoft.com/ntserver/terminalserver/techdetails/prodarch/api.asp

Using and Developing Applications Compatibility Scripts:

http://www.microsoft.com/ntserver/terminalserver/techdetails/prodarch/AppCompSc.asp

A-4 Globalization

Globalization is the practice of designing and implementing software that is not locale dependent, i.e., can accommodate any locale. In software design, a locale is defined as a set of user preferences associated with a user's language. A locale in Windows 2000 includes formats for date, time, currency and numbers; rules and tables for sorting and comparison; and tables of character classifications.

Other user preferences that a globalized application should accommodate include user-interface language, default font selection, language rules for use in spell checking and grammar, and input methods such as keyboard layouts and input method editors. See http://www.microsoft.com/globaldev for more details.

Guidelines for developing a globalized application include the following:

- Use Unicode as your character encoding to represent text. If your application should also run on Windows 9x, consider a method for using Unicode on those platforms, as described in Microsoft Systems Journal articles in the November 1998 and April 1999 issues. These articles are available at:

  http://www.microsoft.com/globaldev/articles/singleunicode.asp
  http://www.microsoft.com/globaldev/articles/multilang.asp

  If you cannot use Unicode, you will need to implement features such as DBCS enabling, BiDi enabling, codepage switching, and text tagging. Guidelines related to this functionality are available at www.microsoft.com/globaldev.
- Consider using a multilingual user interface: launch the application in the default user interface language, and offer the option to change to other languages.
- Use the Win32 API NLS functions to handle locale sensitive data.
- Watch for Windows messages that indicate changes in the input language, and use that information for spell checking, font selection, etc.
- Use the Script APIs (Uniscribe) to layout formatted text on a page. This will allow your application to display multilingual text and complex scripts such as Arabic, Hebrew, Hindi, Tamil, and Thai.
- Test your application on in the following scenarios:
  - Where the system locale, the user locale, the input locale and the UI language each have different values and are all different from the localization language of the application
  - Where the system locale is not English (United States) and not the same as the application localization.
  - Where the user locale is new in Windows 2000 (e.g. Hindi)
  - Where the UI language is one of the right-to-left languages such as Arabic.
  - For a client/server application, where the client and server each have different locales
  - By using Regional Settings to customize the user locale (the more unusual the better for testing)

A-5 Localizability

In contrast to globalization, localization is the process of modifying an application so that its user interface is in the language of the user. Well designed software can be localized to any of the languages supported by Windows 2000 without changes to the source code, i.e., without recompilation. In addition to the guidelines for globalization mentioned above, those for localizability include the following:

- Isolate all user interface elements from the program source code. Put them in resource files, message files, or a private database.
- Use the same resource identifiers throughout the life of the project. Changing identifiers makes it difficult to update localized resources from one build to another.
- Make multiple copies of the same string if it is used in multiple contexts. The same string may have different translations in different contexts.
- Do not place strings that should not be localized in resources. Leave them as string constants in the source code.
- Allocate text buffers dynamically, since text size may expand when translated. If you should use static buffers, make them extra large to accommodate localized strings (e.g., double the English string length).
- Keep in mind that dialog boxes may expand due to localization. Thus, a large dialog box that occupies the entire screen in low-resolution mode may have to be resized to an unusable size when localized.
- Avoid text in bitmaps and icons, as these are difficult to localize.
- Do not create a text message dynamically at runtime, either by concatenating multiple strings or by removing characters from static text. Word order varies by language, so dynamic composition of text in this manner requires code changes to localize to some languages.
- Similarly, avoid composing text using multiple insertion parameters in a format string (e.g., in sprintf or wsprintf), because the order of insertion of the arguments changes when translated to some languages.
- If localizing to a Middle Eastern language such as Arabic or Hebrew, use the right-to-left layout APIs to layout your application right to left. For details, see the article in the April 1999 issue of the Microsoft Systems Journal at http://www.microsoft.com/globaldev/articles/singleunicode.asp.
- Test localized applications on all language variants of Windows 2000. If your application uses Unicode, as recommended, it should run with no modifications. If it uses a Windows codepage you will need to set the system locale to the appropriate value for your localized application, and reboot, before testing.

A-6 Use 64-bit compatible data types

It is possible for developers to use a single source-code base for their Win32- and Win64™-based applications. Microsoft has added this support by introducing new data types in the Platform SDK for Windows 2000.

There are three classes of new data types: fixed-precision data types, pointer-precision types, and specific-precision pointers. These types were added to the Windows environment (specifically, to Basetsd.h) to allow developers to prepare for 64-bit Windows well before its introduction. These new types were derived from the basic C-language integer and long types, so they work in existing code. You can use these data types in your code now, test your code as a Win32-based application, and recompile as a Win64-based application when 64-bit Windows is available

Fixed-precision

Fixed-precision data types are the same length in both Win32 and Win64 programming. To help you remember this, their precision is part of the name of the data type. The following are the fixed-precision data types.

Type	*Definition*
DWORD32	32-bit unsigned integer
DWORD64	64-bit unsigned integer
INT32	32-bit signed integer
INT64	64-bit signed integer
LONG32	32-bit signed integer
LONG64	64-bit signed integer
UINT32	Unsigned INT32
UINT64	Unsigned INT64
ULONG32	Unsigned LONG32
ULONG64	Unsigned LONG64

Pointer Precision

As the pointer precision changes (that is, as it changes from 32 bits with Win32 code to 64 bits with Win64 code), these data types reflect the precision accordingly. Therefore, it is safe to cast a pointer to one of these types when performing pointer arithmetic; if the pointer precision is 64 bits, the type is 64 bits. The count types also reflect the maximum size to which a pointer can refer. The following are the pointer-precision and count types.

Type	*Definition*
DWORD_PTR	Unsigned long type for pointer precision.
HALF_PTR	Half the size of a pointer. Use within a structure that contains a pointer and two small fields.
INT_PTR	Signed integral type for pointer precision.
LONG_PTR	Signed long type for pointer precision.
SIZE_T	The maximum number of bytes to which a pointer can refer. Use for a count that should span the full range of a pointer.
SSIZE_T	Signed SIZE_T.
UHALF_PTR	Unsigned HALF_PTR.
UINT_PTR	Unsigned INT_PTR.
ULONG_PTR	Unsigned LONG_PTR.

Specific Pointer-Precision Types

There are also new pointer types that explicitly size the pointer. Be cautious when using pointers in 64-bit code: If you declare the pointer using a 32-bit type, the system creates the pointer by truncating a 64-bit pointer. (All pointers are 64 bits on a 64-bit platform.)

Type	Definition
POINTER_32	A 32-bit pointer. On a 32-bit system, this is a native pointer. On a 64-bit system, this is a truncated 64-bit pointer.
POINTER_64	A 64-bit pointer. On a 64-bit system, this is a native pointer. On a 32-bit system, this is a sign-extended 32-bit pointer. Note that it is not safe to assume the state of the high pointer bit.

For more details, please see the Microsoft Platform SDK or visit:

http://msdn.microsoft.com/library/psdk/buildapp/64bitwin_410z.htm

A-7 Additional Considerations for Cluster Service

This section identifies addition recommendations, above and beyond the core requirements for applications to support cluster service. These additional recommendations will improve performance and availability.

1. Do not use system registry to store large data structures or data that changes frequently

Server applications may store data in the system registry. The system registry is stored on a local system disk and is not shared or replicated among nodes in the cluster. Cluster Service can replicate system registry keys on behalf of a server application to guarantee that the application, when it moves from one node to another, can always see exactly the same data in the system registry.

To guarantee the consistency of the data, Cluster Service logs any changes to the system registry keys, registered for replication, to one of the shared drives. When a server application fails over to another node, Cluster Service looks for any changes logged and rolls them forward to update the system registry on the current node.

Applications that use the system registry to store large data structures or update it frequently may affect the performance of the cluster and can experience long failover times because of the time needed to roll forward all logged updates.

2. Instrument your application (create an application specific resource DLL) for improved monitoring and failure detection

To protect your application against software failures, such as application crashes, hangs, and slow response time, you should provide a resource DLL that extends Cluster Service capabilities to monitor and detect failures of your application.

3. Support multiple instances of the same cluster resource type

To fully utilize the capacity of all nodes in the cluster, your application should support active/active mode where multiple instances of your application can coexist on the same cluster.

Active/Active failover and failback capability allows two separate instances of a resource type to be running on different nodes, each working with different data sets residing on different disks on the shared SCSI bus. (This means that, although the data isn't shared, the resource type is active on both nodes. By definition, there can only be one instance of a resource; however, there can be multiple instances of a resource *type*.) If the instance of the resource type fails on one node, that instance is moved or "failed over" to the next available node. For example, if your resource type is a database manager application, you can run a copy of the database manager resource type on each node. You can then define a particular database (*db1*) as a resource. With Active/Active capability, you can move *db1* from node 1 to node 2 by telling the database manager on each node to release and acquire the database, as appropriate. This communication cannot happen with generic application or service resource types.

4. Support rolling upgrades of Windows 2000 and the application

Downtime caused by planned operating system and application upgrades can be minimized by performing a *rolling upgrade*.

Rolling upgrade is the process of systematically upgrading each cluster node

while the other nodes continue to provide service. Administrators perform rolling upgrades in stages, first by failing over all groups on a node, taking the node offline, upgrading the node, bringing it back online, failing back the groups to the node, and repeating this process on all other nodes. During a rolling upgrade, services are unavailable only for the time needed to move them from one node to another.

Cluster service supports rolling upgrades of the operating system. However, it's your responsibility to guarantee that your application will function properly during a rolling upgrade. It is also your responsibility to support rolling upgrades of your application.

5. Support '"combo" resource if a large number of instances of the same resource can be installed on the same cluster

Applications that support Active/Active mode allow installing multiple instances of the same resource type. Each instance of a resource uses system resources and contributes to the cluster service overhead. Also, a maximum number of resources supported by a Cluster Service is limited to 1670.

If your application supports Active/Active mode and a cluster can be configured to have a large number of resources of your application type, it is recommended that you design your resource DLL to monitor and detect failures of all instances of your application using a single "combo" resource.

6. Management component of the application should detect cluster installation and use proper cluster APIs to start/stop the application

Once your application is under the Cluster Service control, it should not use Service Control Manager APIs to start/stop services that were configured as cluster resources.

Service Control Manager and Cluster Service do not synchronize their operations. If you used SCM to stop a service, Cluster service would recognize it as a resource failure and would immediately restart the service.

Use **GetNodeClusterState** API to determine if the Cluster Service was installed and is running on a particular node.

Use **ResUtilEnumResources** to find if the particular service is under the control of Cluster Service.

7. Provide cluster-aware setup that can detect cluster installation, install the application on all nodes in the server cluster and configure the cluster

Cluster aware applications are perceived as difficult to install and manage. A cluster ready application should come with a setup program that makes the application installation process as simple as possible and supports following scenarios:

- Complete installation on all nodes in the cluster
- Reliable reinstall and repair on one node
- Install the application on a newly added node
- Reliable and complete uninstall of application on a node evicted from the cluster

Your cluster aware application setup should perform following actions:

- Detect the cluster. Use **GetNodeClusterState**.
- Enumerate all nodes in the cluster. Use **ClusterNodeOpenEnum**, **ClusterNodeEnum**, **ClusterNodeCloseEnum**
- Install the application on all available nodes, schedule installation on nodes currently unavailable. Use **GetClusterNodeState** to determine node state.
- Install the application resource DLL on all nodes and register a new resource type with the cluster using **CreateClusterResourceType** API.
- Register the cluster administration extension DLL with the cluster using **DllRegisterCluAdminExtension** API.
- Publish cluster administrator extension DLLs in Class Store to make it available on any workstation where you want to run Cluster administrator.
- Configure the cluster; add and configure new resources.

It should also meet following requirements:

- Transacted setup: In the event the installation is not completed on all selected nodes, cluster aware setup can roll back to the earlier version of the application without error.
- Support unattended mode.

8. Provide remote administration capability

Your application should support remote administration.

A-8 Windows Management Instrumentation

Windows Management Instrumentation (WMI) is a key component of Microsoft's Windows management services. WMI provides a consistent and richly descriptive model of the configuration, status, and operational aspects of applications and systems. Based on the Distributed Management Task Force (DMTF) Common Information Model (CIM), WMI supports uniform system, device, and applications management in Windows operating systems.

WMI provides a unifying access mechanism to both standard and proprietary instrumentation methods, allowing applications to be managed both as discrete elements and as integrated and inter-related parts of a larger enterprise. By exposing a common access mechanism to all management instrumentation, WMI simplifies the task of developing well-integrated management applications.

Suggested Compliance

1. Access management data through WMI
2. Provide management data to WMI
3. Conform to the rules for extending the CIMV2 namespace
4. Conform to the rules for defining vendor-specific namespaces

Customer Benefits

Customers gain these benefits when products use WMI:

- Management and managed products built on broadly adopted industry management standards.
- Improved ease of use due to a logically organized, consistent model of Windows and Windows application operation, configuration, and status.
- Solutions that manage both local and remote systems and applications transparently through a common, operating system-based management infrastructure.

Implementation Information

This section presents recommendations for creating WMI-compliant products. For the purposes of this section, the use of the terms *application* or *product* mean both user mode applications with a desktop graphical user interface and applications written as one or more services as well as managed products such as hardware devices.

1. Access management data through WMI

WMI provides access to a wide variety of Windows system management information through the CIMV2 namespace, a Win32 extension of the CIM schema. Using WMI, an application has access to various management objects including Win32, Performance Monitor, Registry, Windows Installer service, Active Directory, and Event Log data.

If your product uses management functions and data provided by the system and/or other products, and that data is available in the CIMV2 namespace, your product should use WMI to access that management information rather than native APIs.

2. Provide Management data to WMI

If your product has management functions and data that permit control or monitoring of your product by other management applications, it should be exposed through WMI according to the following recommendations (see the WMI

SDK for development details and schema extension rules http://msdn.microsoft.com/developer/sdk/wmisdk):

- Use or extend the existing CIM classes in the CIMV2 namespace or create another namespace with new classes that represent the management functions and data.
- Rules for extending CIMV2 are documented in the WMI SDK (formerly called the WBEM SDK).
- In cases where it is not appropriate to add new classes or instances to CIMV2 (as defined by the SDK documentation), a new namespace should be created.
- Provide instances for the newly defined classes. This can be done using static instances defined in a Managed Object Format (MOF) file or dynamic instances generated at runtime by a WMI Provider. If instances are provided through an MOF file, the vendor is responsible for ensuring the MOF file is compiled by the WMI MOF compiler during product setup.
- If the product generates events, an event Provider should be implemented. The event Provider sends notifications of events to WMI, which then forwards the events to applications registered to receive those events. The implementation guidelines for event Providers are included in the WMI SDK documentation.

3. Rules for extending the CIMV2 namespace

If your product extends the CIMV2 namespace, it should conform to the following guidelines.

Notes:

- Different types of data have different extension rules.
- If two subclasses are derived from a class, and if a particular instance is generated by providers of both the subclasses, then WMI will accept one of them and log an error for the second one. In order to avoid such conflicts, subclassing should be done only if you own the instance of the subclass. Accordingly, it is recommended that subclassing should be done only by the manufacturer of the entity that the instance represents.

Application Data

The following classes and instances are created automatically when an application is installed using the Windows Installer service. Refer to the 'Settings Data' section below if you are modeling configuration information not created automatically by the Windows Installer Provider:

- The application is represented by an instance of **Win32_ApplicationService** (subclassed from **CIM_Service**) representing the presence (either advertised or installed) of the application on the system.
- The command line used to execute the application is represented by an instance of **Win32_CommandLineAccess** (subclassed from **CIM_ServiceAccessPoint**). These are associated with the instances of Win32_ApplicationService using the Win32ApplicationCommandLine association (subclassed from CIM_ServiceAccessBySAP).
- The primary components of the application are represented by instances of **Win32_SoftwareElement** (subclassed from **CIM_SoftwareElement**). Some examples of primary components are executables, DLLs, and database files.
- Application installation configuration is represented by instances of objects subclassed from **CIM_Setting**. These are associated with the installed **Win32_SoftwareElements** using the **Win32_SoftwareElementResource**

association.

- Installation configuration of the open database connectivity (ODBC) driver is represented by subclasses of the **Win32_SettingCheck** association.
- If the application has any configuration information, the information is visible as a subclass of the **CIM_Setting** class and should be related to the correct instance of the **Win32_ApplicationService** (that represents the application on the system). The **CIM_Setting** instance is associated with the **Win32_ApplicationService** instance using a subclass of the **CIM_ElementSetting** association class. Note that there may be multiple instances of the CIM_Setting class representing the possible configurations. The configuration information is settable via the CIM_Setting class. However, the current configuration information is typically stored in the Win32_ApplicationService instance.

Settings Data

The rules for extending the schema for settings (for example, any settable parameter defined for services, systems, applications or devices) are:

- The setting should be expressed through a subclass of the **CIM_Setting** class. Note that there may be multiple instances of the CIM_Setting class representing the possible configurations. The configuration information is settable via the CIM_Setting class. However, the current configuration information is typically stored in the CIM_ManagedSystemElement descendent instance.
- An instance of a subclass of the **CIM_ElementSetting** association should be defined to establish a relationship between **CIM_Setting** and the corresponding descendant of **CIM_ManagedSystemElement** (the service, system, application, or device to which the setting applies).

Statistics Data

The rules for extending the schema for statistics are:

- Any statistics information about a descendant of **CIM_ManagedSystemElement** should be expressed through a subclass of the **CIM_StatisticalInformation** class or be included as property data in the object itself.

  Where the data is closely coupled with the Managed System Element and should be retrieved as native properties of the object, the data is placed as properties of the object itself (for example, packets transmitted or received on an Ethernet adapter).

  Alternately, where the data may not always be available, is derived from existing data or is not required to manage the object, it should be placed in a subclass of **CIM_StatisticalInformation** (for example, major, minor, or warning error counts on a device).
- The vendor should define an instance of a subclass of the **CIM_Statistics** association to establish the relationship between **CIM_StatisticalInformation** and the corresponding descendant of **CIM_ManagedSystemElement**.

Device Data

The rules for extending the schema to represent a device are:

- Vendor-specific classes should derive from leaf-most subclasses of CIM_LogicalDevice.
- If the device has configuration information, the information should be visible as a subclass of CIM_Setting and should be related to the correct instance of the CIM_LogicalDevice subclass (that represents the device in the system)

using a subclass of the CIM_ElementSetting association class. Note that there may be multiple instances of the CIM_Setting class representing the possible configurations. The configuration information is settable via the CIM_Setting class. However, the current configuration information is typically stored in the CIM_ManagedSystemElement descendent instance.

- Do not subclass directly from CIM_LogicalDevice. The WMI SDK contains details of the classes that may be subclassed and instantiated.

Computer System Data (typically applies to OEMs)

The rules for extending the schema to represent a computer system are:

- The vendor should define a direct subclass from **Win32_ComputerSystem**.
- Do not subclass directly from **CIM_UnitaryComputerSystem**.

Physical Configuration Data

The rules for extending the schema to represent physical configuration:

- Vendor specific classes should derive from leaf-most subclasses of **CIM_PhysicalElement** unless Microsoft provides a **Win32** schema class for the particular Physical Element. In the latter case, the vendor specific classes should derive from that **Win32** class.

Pretesting Guidelines

Microsoft is developing tools that will help to automate the verification process for compliance to the WMI recommendations. Please check http://msdn.microsoft.com/developer/sdk/wmisdk/ for updated information. In the interim, we recommend using CIM Studio to verify the correctness of the schema design. CIM Studio is included with the WMI SDK. To use CIM Studio for pretesting, make sure you have:

- Installed the WMI SDK
- Created the MOF files with your schema extensions, where applicable
- Compiled the schema into the repository using mofcomp.exe, where applicable
- Connected to root\CIMV2 namespace
- Located the classes you have added to the namespace (Click on the Find button to invoke the search facility)

To pretest correctness of schema extension for application data:

1. Make sure that your product is installed using the Windows Installer service. This will guarantee correct extension of the CIMV2 namespace for the product data.
2. If your product has configuration information, you should pretest your Settings Data

To pretest correctness of schema extension for settings data:

1. Make sure that your class is derived from the **CIM_Setting** class directly or a leaf-most subclass of **CIM_Setting**.
2. Make sure you have defined a subclass of the **CIM_ElementSetting** association that establishes relationship between **CIM_Setting** and the descendant of **CIM_ManagedSystemElement** that you have defined for the service, system, application, or device to which the setting applies.
3. Make sure that your Provider provides dynamic instances of the classes you have defined.

To pretest correctness of schema extension for statistics data:

1. If the statistical data is closely coupled with your Managed Element and should always be retrieved as native properties of the object, make sure that you include the data as properties of the subclass of **CIM_ManagedSystemElement**.
2. If the statistical data is not always available, is derived from existing data, or is not required to manage the System Element, make sure that you have defined a subclass of **CIM_StatisticalInformation**.
3. Make sure you have defined a subclass of the **CIM_Statistics** association that establishes a relationship between **CIM_StatisticalInformation** and the descendant of **CIM_ManagedSystemElement** for the service, system, application, or device to which the statistics applies.
4. Make sure that your Provider provides dynamic instances of the classes you have defined.

To pretest correctness of schema extension for device data:

1. Make sure you have defined a subclass of a leaf-most descendant of **CIM_LogicalDevice**.
2. Make sure you do not subclass directly from **CIM_LogicalDevice**.
3. If the device has configuration information, you should pretest your Settings Data.
4. Make sure that your Provider provides dynamic instances of the classes you have defined.

To pretest correctness of schema extension for system data:

1. Make sure you have defined a direct subclass of **Win32_ComputerSystem**.
2. Make sure that your Provider provides dynamic instances of this class.

To pretest correctness of schema extension for physical configuration data:

1. If Microsoft provides a Win32 schema class for the particular Physical Element used, make sure that you have defined a subclass of it.
2. If Microsoft does not provide a class for the particular Physical Element used, make sure you have defined a subclass of a leaf-most descendant of **CIM_PhysicalElement**.
3. Make sure that your Provider provides dynamic instances of this class.

A-9 Microsoft Management Console (MMC)

MMC is an extensible, common console designed to integrate management tools and functions, and to present a common visualization environment for management applications. Management applications should be designed to take advantage of the features and support provided by MMC to enable them to present a common interface to the user.

Integrating the look and feel of management functionality in this way provides greater efficiency to administrators, who often combine a number of different management tools to solve a particular problem. Having these tools operate in a similar manner is very helpful when managing an enterprise environment.

MMC also reduces the time to market for developers of management products by providing the foundation for the user interface of those applications. Developers can extend existing Console functionality or provide new functionality by the creation of one or more Console snap-ins. Snap-ins are the means by which new features and application behavior are added to the Console.

Suggested Compliance

1. Extend the Computer Management and/or the Users and Groups Snap-Ins with a context menu item that conforms to the MMC user interface guidelines for context menu item.
2. Write an MMC namespace snap-in.

Customer Benefits

System administrators in corporate environments gain these benefits when your application uses MMC:

- **Reduced learning time**: A simpler interface and common look and feel for applications helps administrators learn your product more quickly, maximizing the benefit of the product in the shortest time. The Microsoft BackOffice® suite of products uses MMC to give its management-related features a Windows-based visual interface, providing administrators with a familiar interface so they can quickly learn new applications and information.
- **Logical grouping of functionality**: MMC groups management functions into common categories, making it easier for administrators to find the right tool for the task.
- **Integration of management tools**: MMC provides the foundation for the integration of management tools that previously operated in complete isolation of each other. Integrating functionality in this way provides the administrator with the means to solve problems and manage the environment in a more efficient and time sensitive-manner.
- **Task delegation**: MMC allows administrators to define and distribute specific sets of management functionality to other peer or subordinate administrators, removing the clutter of unnecessary application functionality that doesn't pertain to the immediate task.
- **Leveraging console snap-ins**: Custom corporate and third-party applications can take advantage of existing snap-in functionality to minimize the development time of their applications, while at the same time maintaining the advantages of a common look to those applications.

Implementation Information

1. Extend the Computer Management and/or the Users and Groups Snap-Ins with a context menu item that conforms to the MMC user interface requirements for context menu items

The simplest way to integrate your application with MMC is to add a context menu item to launch your standalone management tool from one or both of the core Windows 2000 snap-ins. Beyond this, tighter integration is achieved by writing a namespace extension to one of these snap-ins. The section below shows you how to create a context menu extension. Complete information on namespace extensions is available in the MMC SDK section of the Microsoft Platform SDK.

1. Decide which node you need to extend. The Computer Management root node should not have its namespace extended, but it may have its context menu or property pages extended. The example below assumes you are adding a context menu item to the Computer Management root node. If you want to extend a different node, you should use its GUID in place of the one in the following example.
2. Register your snap-in as described in the MMC section of the Microsoft Platform SDK. In addition, add a value named with your snap-in's GUID under the following registry key:

   HKLM\Software\Microsoft\MMC\Nodetypes\
   {476e6446-aaff-11d0-b944-00c04fd8d5b0}\Extensions/ContextMenu

3. Create a snap-in which implements **IExtendContextMenu()**. This snap-in will invoke your standalone tool when the context menu is selected. Place the snap-in at the CCM_INSERTIONPOINTID_3RDPARTY_TASK insertion point. See the MMC SDK documentation in the Microsoft Platform SDK for full details on insertion points. You may find it easier to use the Microsoft Visual C++® 6.0 ATL wizard for MMC.

If you extend the Users and Groups snap-in rather than the Computer Management snap-in, you will have to use the class GUID for the node you extend. The class GUID is found in the Directory Service section of the Microsoft Platform SDK. Use the Directory Service class registration tool found in this section of the SDK and follow the steps listed earlier in this requirement to replace the Computer Management GUID with the correct one from the Directory Service section.

2. Write an MMC namespace snap-in

Microsoft recommends that you write an MMC namespace snap-in as an alternative to extending a context menu. A namespace snap-in offers the greatest value to an MMC user by taking full advantage of MMC features.

If you write a full namespace extension snap-in, follow the guidelines presented in the MMC snap-in author's guide. These guidelines ensure that snap-ins from different products operate consistently.

This guide can be found at http://www.microsoft.com/management/mmc.

Pretesting Guidelines

To pretest a snap-in:

- Standalone Mode.

  If the snap-in runs in a standalone mode, then load the snap-in in a user-mode Console and ensure the snap-in performs as expected.
- Extends Computer Management.

If the snap-in extends Computer Management, then start the Computer Management console file from the Start menu and verify that your snap-in performs as expected.

- Extends Users and Groups Snap-In

  If the snap-in extends the Users and Groups snap-in, then start the Users and Groups console file from the Start menu and verify that your snap-in performs as expected.

- Context menu extension.

  If your snap-in is a context menu extension, then ensure that you do not add separators, that cascading menus are used sparingly and only on the New and Task menus, and that your items appear in the desired order Appendix.

A-10 Expose a COM-based scripting model

All applications should provide a COM-based scripting model, which will allow system administrators and integrators to access the functionality of the application or its configuration information programmatically. The scripting model should be accessible from the Windows Scripting Host and Active Server Pages.

Suggested Compliance

- Applications should provide one or more COM components that expose the functionality of the application or its configuration information.
- Components should expose Automation (i.e. IDispatch-based) interfaces or dual interfaces.
- Applications should provide programmer's documentation describing how to use the scripting model.

Customer Benefits

By providing a scripting model, you enable your customers to:

- Automate routine configuration and administration tasks
- Develop customized user interfaces

Implementation Information

- *Platform SDK, COM and ActiveX Object Services, Automation* describes Automation and how to implement COM components that expose Automation interfaces in C or C++.
- *"The Basics of Programming Model Design",* Dave Stearns, MSDN Library describes how to define the scripting model.
- *"Building COM Components That Take Full Advantage of Visual Basic and Scripting",* Ivo Salmre, MSDN Library describes how to define scripting models, with special emphasis on access from Visual Basic and scripting languages.
- *Visual Basic Component Tools Guide, Creating ActiveX Components* describes how to implement COM components that expose Automation interfaces using Visual Basic.

A-11 Use COM+ for Distributed Applications

Your server-based applications should be configured as COM+ applications to take advantage of the benefits provided by COM+. Use the Component Services snap-in for MMC to add your application to the COM+ Applications folder for the computer it is installed on.

Compliance Guidelines

COM+ Applications are created from one or more components that use COM+ services and are installed in the COM+ catalog.

- All components should be implemented in a Dynamic Linking Library (DLL)
- Each component should expose one or more Automation (i.e. IDispatch-based) or dual interfaces.
- Use the Component Services snap-in for MMC or the COM+ Administration APIs to register each COM component as part of a single COM+ application.
- Use the Component Services snap-in for MMC or the COM+ Administration APIs to export the COM+ application.
- Use the Component Services snap-in for MMC or the COM+ Administration APIs to export information required by remote clients to access the COM+ application.
- Provide documentation for each COM component.
- Describe the syntax and expected results for every method on every interface intended for use by developers or administrators.

Customer Benefit

COM+ defines a *general purpose* programming model for building distributed component-based server applications. Developers architect, design, and build their application servers as *COM+ Components* to be hosted in *COM+ Applications* that in turn leverage the Windows 2000 *Component Services* run-time (also known as the COM+ Services run-time). This execution environment provides the high scalability, robustness, and integrity traditionally associated only with high-end transaction processing systems. The COM+ execution environment automatically and transparently handles the complexities and details of transaction management including: synchronization of shared resources, process and thread management, context management, role-based security and load balancing.

A COM+ application is a set of one or more in-process components installed in the COM+ catalog. COM+ applications are started on demand, without user intervention. The catalog can contain logon and security information for an application so that it can run even though no user is logged on. System administrators can also customize security requirements for each application and thereby control access to it.

COM+ applications may be installed and controlled using standard user and Win32 APIs. Applications can be controlled both locally and remotely through the Component Services administrative tool, providing network administrators an easy and consistent way to control the application across the network.

The Platform SDK provides more information on building and deploying COM+ Applications.

Implementation Information

- See the Microsoft COM+ website for more information on developing COM+ applications: http://www.microsoft.com/com.

- The *Component Services Programmer's Guide* describes how to implement COM+ User Events.
- *Platform SDK, COM and ActiveX Object Services, Automation* describes Automation and how to implement COM components that expose Automation interfaces in Microsoft® C or C++ development systems.
- *"The Basics of Programming Model Design,"* Dave Stearns, MSDN Library describes how to define the scripting model.
- *"Building COM Components That Take Full Advantage of Visual Basic and Scripting,"* Ivo Salmre, MSDN Library describes how to define scripting models, with special emphasis on access from Visual Basic and scripting languages.
- *Visual Basic Component Tools Guide, Creating ActiveX Components* describes how to implement COM components that expose Automation interfaces using Microsoft Visual Basic®.

A-12. Run without NetBIOS in Windows 2000-only environment

It is possible to configure a Windows 2000-only environment to run without NetBIOS, and without the NetBIOS name service (i.e. the Windows Internet Name System - WINS).

Therefore, to ensure that applications can run in a Windows 2000-only environment:

- Applications must be able to accept as user input both NetBIOS and DNS names for computers and domains.
- Applications that run only on the Windows 2000 platform and obtain names programmatically must always request the DNS version of the computer or domain name.

Definitions of maximum computer name length and name validation APIs are discussed in the *Active Directory Programmer's Guide*.

Working with Computer Names

For applications that run exclusively on the Windows 2000 platform:

- Call **GetComputerNameEx** and request ComputerNameDnsFullyQualified to obtain the local computer name. This API supercedes **GetComputerName** on the Windows 2000 platform.
- Call **SetComputerNameEx** and pass ComputerNameDnsHostname or ComputerNameDnsDomain as appropriate to set the local fully qualified computer name. This API supercedes **SetComputerName** on the Windows 2000 platform.

For applications designed to run on all platforms which handle computer names in Unicode format, the **WSALookupService** family of APIs should be used for computer name resolution instead of **gethostbyname**.

Working with Domain Names

To resolve domain names to Domain Controller names:

- Call **DsGetDcName** with flag DS_PDC_REQUIRED to obtain the name of the Primary Domain Controller for a given domain. This API supercedes **NetGetDCName** on all platforms.
- Call **DsGetDcName** to obtain the name of any Domain Controller for a given domain. This API supercedes **NetGetAnyDCName** on all platforms.

Appendix B

Browser-Hosted Applications

As described in requirement 1.10 in this Server Specification, clients that ship as part of distributed applications must comply with the Desktop Application Specification for Windows 2000 in order for the application to be certified. However, special considerations apply for browser-hosted applications:

- Clients that are browser-hosted Web pages that use only pure HTML and DHTML and contain no executable content (e.g. no ActiveX or Java controls) need not be tested for full compliance with the Desktop Application Specification, as they will for the most part inherit the compliance of their host browser. However, these browser-hosted clients are required to meet the following two specific requirements:
  - Provide documented keyboard access to all features
  - Do not rely exclusively on sound.

  For information on designing accessible Web content see "Web Content Accessibility Guidelines" at http://www.w3c.org/WAI/.
- Browser-hosted clients that include executable content, such as ActiveX or Java controls, must comply with the requirements in the Desktop Application Specification indicated on the checklist below.

  Note that some requirements below may not apply to your application if your application does not expose functionality covered by the requirement. For example if your application does not expose any file names to the user, then requirement 1.3 does not apply.

	Windows Fundamentals
X	1.1 Perform primary functionality and maintain stability
X	1.2 Provide 32-bit components
X	1.3 Support Long File Names and UNC paths (*if you expose file names*)
X	1.4 Support printers with long names and UNC paths (*if you support printing*)
X	1.5 Do not read from or write to Win.ini, System.ini, Autoexec.bat or Config.sys on any Windows operating system based on NT technology.
X	1.6 Ensure non-hidden files outside of your application directory have associated file-types, and all file-types have associated icons, descriptions, and actions
X	1.7 Perform Windows version checking correctly (*if you need to determine the Windows version*)
X	1.8 Support AutoPlay of compact discs *(if you distribute your product on CD*)
X	1.9 Kernel mode drivers must pass verification testing on Windows 2000 (*if you have any*)
X	1.10 Hardware drivers must pass WHQL testing (*if you have any*)

	Install/Uninstall
X*	2.1 Install using a Windows Installer-based package that passes validation testing
X*	2.2 Observe rules in componentization
X*	2.3 Identify shared components
	2.4 Install to Program Files by default
	2.5 Support Add/Remove Programs properly
X*	2.6 Ensure that your application supports advertising
	2.7 Ensure correct uninstall support

** ActiveX controls must be packaged using MSI in order to run in a secure environment. See Knowledge Base Article Q241163*

	Component Sharing	
X	3.1	Do not attempt to replace files that are protected by System File Protection
	3.2	Component producers: Build side-by-side components
	3.3	Application developers: Consume and install side-by-side components
X	3.4	Install any non side-by-side shared files to the correct locations
	Data and Settings Management	
X	4.1	Default to My Documents for storage of user-created data (*if you have local file I/O*)
X	4.2	Classify and store application data correctly
X	4.3	Degrade gracefully on access denied
X	4.4	Run in a secure Windows environment
X	4.5	Adhere to system-level Group Policy settings
X	4.6	Applications that create ADM files must properly store their ADM file settings in the registry *(if you supply ADM files)*
	User Interface Fundamentals	
X	5.1	Support standard system size, color, font, & input settings
X	5.2	Ensure compatibility with the High Contrast option
X	5.3	Provide documented keyboard access to all features
X	5.4	Expose the location of the keyboard focus
X	5.5	Do not rely exclusively on sound
X	5.6	Do not place shortcuts to documents, help, or uninstall in the Start Menu
X	5.7	Support multiple monitors *(if your executable code specifies screen coordinates you need to verify that it does so properly)*
	OnNow/ACPI Support	
	6.1	Indicate busy application status properly
	6.2	Respond to sleep requests from the operating system properly
	6.3	Handle sleep notifications properly
	6.4	Handle wake from normal sleep without losing data
	6.5	Handle wake from critical sleep properly
	Application Migration	
N/A	7.1	Application must continue to function after upgrade to Windows 2000 Professional without reinstall

Appendix C

Usage of Non-Compliant 3rd Party Components

In some cases, and for a limited time, your application may be eligible for Certification even if it relies on non-compliant third-party database or message store components.

This exemption applies only when all of the following requirements are met:

- The only non-compliant components are database and/or message store components.
- Your organization does not have access to or permission to modify the source for these components.
- Your server application requires these components in order to run.
- You *submit* your application to VeriTest for server certification prior to Feb 1, 2001.
- All other components of your application *pass* certification testing by April 1, 2001.
- If your application has a readme, you must document the fact that your product requires non-certified components in that readme and you must provide a link to the VeriTest website where your Certification Results are hosted, which is currently http://www.veritest.com/mslogos/windows2000/certification/. If your application has no readme, then you must document this prominently in your product documentation.

Important Note: Databases and message stores are the only components subject to this exemption.

Appendix D

Summary of Changes

Summary of changes from spec 1.19 draft to 1.2 Specification

General

- Added intro: “Who is this Specification for”
- Added requirement 1.10: “Clients of distributed applications must comply with the Desktop Application Specification” to clarify the content that was previously covered in the note “Client and Server as used in this specification.”
- Added Appendix B “Browser-hosted applications”
- Added Appendix C “Usage of non-compliant third party components”

Install

- Dropped requirement 2.1: “Check for access and availability of resources during install.” This is generally adopted today as a good practice and we believe this will be done without putting it in the Specification. This will save testing costs.
- Clarified the “Do not overwrite files with older versions” to be specific to no proprietary files and shared files.

ActiveDirectory:

- Rewrote requirement 4.1: This is now “Use Active Directory appropriately.” We focus on 2 scenarios: publishing to Active Directory to enable binding between client and server components, and using information in Active Directory.
- Replaced requirement 4.2. This is now “Document the storage and replication impact of your application.” Previous requirement was “Do not store data in Active Directory that changes more often than once every 48 hours on a regular basis.” This rewrite better addresses the varying needs of enterprises and equips them with the information to evaluate the replication impact of your application in their environment.
- Added requirement: “Document your application’s use of objects and attributes in the base schema.”
- Modified requirements size limitations to include objects as well as attributes.
- Updated the schema naming convention to focus only on company name instead of DNS name. Microsoft will provide a registration service for these names to ensure uniqueness.
- Extended exemption to June 1, 2000.
- Added Best Practice: Run without NetBIOS in Windows 2000-only environment
- Added Pre-Test section

Security:

- Clarifications on implementing Single Sign-On.

Summary of changes from spec 1.18 to 1.19 draft

- Added intro section: “Client and Server as used in this specification”
- Rewrote requirement 1 in Active Directory chapter.
- Refined requirement 1 in Security chapter regarding documentation.
- Editing throughout.

Summary of changes from spec 1.1 to 1.18 draft

Active Directory Chapter:

- Rewrote chapter to allow more flexibility in the deciding when to publish or locate info from active directory. The previous requirements 4.1 and 4.2 were folded into one requirement which is now 4.1.
- Added 3 requirements to provide guidance as to what type of information should go into AD. To preserve network bandwidth, data that is stored in AD must be reasonably small, must not change often too large, and must be globally interesting.
- Modification to rules for extending the schema –
  - Changed the naming convention to focus on DNS name only. The year of registration is no longer required as part of the attribute name.
  - An exemption will be allowed for applications that have already established naming conventions, provided these naming conventions are registered with MS by March 1, 2000.
  - Added requirement regarding specification of Link-ID

Clustering chapter.

- Rewrote requirements to focus on testable behavior. Many of the previous requirements were prescriptive guidelines to help meet the ultimate requirements of properly handling failover. These are still included as guidelines to help developers meet the core testable requirements. While there are fewer guidelines, this does not represent a softening of the spec. Rather we want to focus on what will be tested.
- Added pre-test steps

Security Chapter.

- Requirement 5.1 (“Configure the server to run under the appropriate service account”) was rewritten to be clearer. It is now “Document services that require more than User level privileges to run.”
- Requirement 5.2 (“Connection Authentication using client credentials”) was rewritten to more clearly focus on single sign-on.
- Requirement 5.3 (Impersonation of the Client”) was made a recommendation. Issue is that for applications that have server private data, it is not always feasible or appropriate to impersonate the client.
- Rewrote pre-test steps

Install Chapter:

- Renamed the requirement “Check access to and availability of resources before install” to “Before install, check for file system access, sufficient disk space, and registry access”

Summary of changes from spec 1.0 to the 1.1 draft.

- Reorganized the chapters to focus on server apps. Client portions (except for those in the Security, Active Directory, and Clustering chapters) eliminated, since these are covered in the desktop spec.
- We are postponing the requirement to use Windows Installer for server based applications until we can provide greater support in the Windows Installer to natively support select server needs. We updated the install chapter to reflect this, and absorbed relevant info from the old "Component Sharing" into this new chapter. Use of the Windows Installer is still appropriate for many server applications, thus it is strongly recommend that vendors consider using it now. It is still required for desktop applications and clients of client/server apps that wish to be certified.
- Dropped OnNow requirement for server apps. Relevant only for desktop applications.
- Dropped the Active Directory requirement to support both NetBios and DNS names. Issue is enabling this functionality on downlevel platforms is extremely difficult.
- Dropped the Migration chapter as this was for desktop apps.
- Dropped the "Data and Settings Mgmt" chapter since this is desktop focused. Not relevant for server applications.
- Dropped multi-monitor req,
- Dropped the "don't rely on sound" accessibility requirement.
- Clarified the Long file names requirement
- Clarified the long printer names requirement. Win2000 support long printer names up to 220 characters.

Best Practices:

- Dropped the "Additional Considerations for Accessibility"
- Updated the WMI section.

Glossary

Active Accessibility A COM-based mechanism that allows applications to actively cooperate with software tools running in the system, such as automation tools, testing tools, and accessibility aids used by people with disabilities.

Active Directory Provides the ability to build applications that give a single point of access to multiple directories in a network environment, whether those directories are LDAP, NDS, or NTDS based directories.

CSIDL These constants provide a unique system-independent way to identify special folders. Used in conjunction with SHGetFolderPath and other APIs.

Degrade gracefully Does not crash the application or the operating system (GPF or blue screen), and does not lose user data. A dialog box or other visual and audio cue appears informing the user, for example, that the functionality is not available on *X* version of *Y* operating system. User is not required to close the application, and can continue to use the other functionality.

Down-level operating system Any combination of these three operating systems: Windows 95, Windows 98, Windows NT Workstation 4.0, or Windows NT Server 4.0.

Group Policy Used to specify settings for groups of users and computers, including software policies, scripts, user documents and settings, application deployment, and security settings.

High Contrast support An option set by the user indicating that they require a high degree of contrast to improve screen legibility. Some application features may be exempted, such as when the use of color is intrinsic and indispensable to the goal of the feature.

HKCU short for HKEY_CURRENT_USER

HKLM short for HKEY_LOCAL_MACHINE

IntelliMirror A set of management technologies that mirror systems, data, and applications on a server. Part of the Zero Administration initiative for Windows (ZAW).

long file name (LFN) Any filename that exceeds 8.3 characters in length or contains any character that is not valid in the 8.3 namespace.

multi-master replication. In Active Directory, means that all replicas of a given partition are writeable. This allows updates to be applied to any replica of a given partition. The Active Directory replication system propagates the changes from a given replica to all other replicas. Replication is automatic and transparent.

resource

Resource (wrt clustering). A physical or logical entity that is capable of being owned by a node, brought online and taken offline, moved between nodes, and managed as a server cluster object. A resource can only be owned by a single node at any point in time.

Secure Windows Environment A configuration that prevents unprivileged Users from intentionally or accidentally compromising the operating system. This is defined as the environment on Windows 2000 exposed to a normal (non-admin\non-power) User by default on a clean-installed NTFS system. In this environment, Users can only write to three* specific locations:

1. Their own portion of the registry (HKEY_CURRENT_USER)**
2. Their own user profile directory (CSIDL_PROFILE)
3. A Shared Documents location (CSIDL_COMMON_DOCUMENTS)***

Users have Read-Only access to the rest of the system.

*Applications are free to modify the default security for an application-specific subdirectory of CSIDL_COMMON_APPDATA provided the modification is documented in the vendor questionnaire. This may provide a fourth location for Users to write to for a given application.

**Users can not write to the following sections of HKCU:

\Software\Policies

\Software\Microsoft\Windows\CurrentVersion\Policies

***By default, Users cannot write to other Users shared documents, they can only read other Users shared documents. Applications are free to modify this default security on an application-specific subdirectory of CSIDL_COMMON_DOCUMENTS provided the modification is documented in the vendor questionnaire.

Side-by-side sharing A new form of sharing in Windows 2000 and Windows 98 Second Edition that enables multiple versions of the same DLL to run at the same time.

System caret Normally the flashing vertical bar that indicates the insertion point in text, it can actually have a range of appearances and is used to indicate keyboard focus location to other software utilities.

trusted domain account An account that is in the same domain as the machine that the service is running on or in a domain which that domain trusts.

universal naming convention (UNC) The system for indicating names of servers and computers, such as \\Servername\Sharename.

user profile A computer-based record maintained about an authorized user of a multi-user computer system. A user profile is needed for security and other reasons; it can contain such information as the user's access restrictions, mailbox location, type of terminal, and so on.

Windows Installer service Provides end users with a way to install and remove applications, or components of software as needed. System administrators can more easily manage applications and support roaming users.