Driver Installation for 32-bit and 64‑bit Platforms

July 21, 2008

Abstract

If your device will be installed on both 32-bit platforms and 64-bit platforms, you must provide a native driver package for each platform. You can, however, use a single 32-bit installer to install either driver package. This paper provides information about how use a 32-bit installation program to install either 32‑bit or 64‑bit drivers for the Windows® family of operating systems.

This information applies for the following operating systems:
 Windows Server® 2008
 Windows Vista®
 Windows Server 2003
 Windows Server 2003 64-Bit Edition
 Windows XP
 Windows XP 64-Bit Edition

For up-to-date information, see “Device Installation Design Guide” in the WDK documentation at:

<http://msdn2.microsoft.com/en-us/library/ms790231.aspx>

The current version of this paper is maintained on the Web at:
 <http://www.microsoft.com/whdc/driver/install/32-64bit_install.mspx>

References and resources discussed here are listed at the end of this paper.

Disclaimer: This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place or event is intended or should be inferred.

© 2005–2008 Microsoft Corporation. All rights reserved.

Microsoft, Windows, Windows NT, Windows Server, and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

#### Revision History

|  |  |
| --- | --- |
| Date | Change |
| 7/21/2008 | Resource links updated |
| 7/15/2008 | Resource links modified per Bug# 434 |
| 12/17/2007 | Update to WDK documentation references |
| 3/25/2005 | Created |

Contents

[Introduction 4](#_Toc203979393)

[Installation Considerations for 64-bit Platforms 4](#_Toc203979394)

[Driver Installation Packages for 32-bit and 64-bit Platforms 5](#_Toc203979395)

[Driver Packages 5](#_Toc203979396)

[DPInst Installation Packages 6](#_Toc203979397)

[Driver Package Installers 6](#_Toc203979398)

[MSI Installation Packages with DIFxApp 7](#_Toc203979399)

[Examples 7](#_Toc203979400)

[Launching the Correct Version of DPInst from an Installer 7](#_Toc203979401)

[Launching the Correct Version of DPInst from a Command Script 11](#_Toc203979402)

[Resources 11](#_Toc203979403)

# Introduction

To simplify installation of 64-bit drivers from a 32-bit installer, Microsoft is providing 64-bit versions of the Driver Install Frameworks (DIFx) tools in the Windows Server® 2003 SP1 DDK. The DIFx tools include:

* The Driver Package Installer (DPInst)
* The Driver Installation Frameworks for Applications (DIFxApp)

Instead of writing a separate 64-bit installer for 64-bit platforms, you can launch the correct version of DPInst or DIFxApp from a single 32-bit installer.

32-bit driver installation packages cannot be used to directly install 64-bit drivers, but they can be used to determine which architecture is being run and then launch the appropriate installer. On 64-bit platforms, the 32-bit installer runs under Windows® 32-bit on Windows 64-bit (WOW64), the x64 emulator that allows Win32®-based applications to run on 64-bit Windows. The installer must be able to detect the platform on which it is running so it can install the correct driver package for that platform.

This paper provides information about how to install either 32-bit or 64-bit drivers for the Windows family of operating systems from a 32-bit installation program.

For information about the DIFx tools, see Resources at the end of this paper.

# Installation Considerations for 64-bit Platforms

The 64-bit version of Windows is designed to make it possible for developers to use a single source code base for Win32 and Win64 applications. To a large extent, this is also true for 32-bit and 64-bit Windows drivers.

For user-mode applications, 64-bit Windows includes an emulation layer called WOW64 that enables Win32 applications to execute on Win64 systems. WOW64 does this by intercepting Win32 function calls and converting pointer-precision parameter types to fixed-precision types as appropriate before making the transition to the 64-bit kernel. (This conversion process, which is sometimes called thunking, is done only for 32-bit applications; 32-bit drivers are not supported on 64-bit Windows.)

Installation functions such as UpdateDriverForPlugAndPlayDevices, InstallSelectedDriver, and SetupCopyOEMInf are not WOW64-capable and will return an error if called from a 32-bit process on 64-bit Windows. This functionality was not enabled for the following reasons:

* The DIFx tools are the preferred method for driver installation on Windows. These tools are available in 64-bit versions, and they will evolve in functionality and versatility in future versions.
* It was not possible to enable legacy 32-bit installers to work on 64-bit Windows simply by enabling installation functions for WOW64. At a minimum, this would have required a new INF model and significant changes to many legacy installers.
* The computing model is evolving toward 64-bit platforms. Creating a framework and set of rules that would allow 32-bit installation packages to work with 64-bit drivers would only offer interim solutions.

A 32-bit application, including a device installer, can launch a 64-bit executable by calling CreateProcess. WOW64 can intercept this call and make a successful transition to the 64-bit operating system.

# Driver Installation Packages for 32-bit and 64-bit Platforms

A driver package consists of the files that are required to install and service a device on a given Windows platform. A driver installation package consists of one or more driver packages plus a tool such as DPInst that installs the driver package.

If a driver installation package contains only a 32-bit driver package, the installer should display an informative error message if the user attempts to install the package on a 64-bit platform. For example, an installer should provide a message that is similar to the following message that the Found New Hardware Wizard displays:

Windows cannot find a compatible software driver for this device. If a driver was provided, make sure it is designed to work with Windows for x64-based systems. If you do not have an appropriate driver, contact the device manufacturer.

## Driver Packages

A driver package targets one or more platforms. For example, a driver package can target only x86 or AMD64, or it can target both x86 and AMD64. It can also target Intel Itanium-based systems, but for simplicity this paper focuses on x86 and AMD64 because x86-based applications and installers can also run on AMD64.

When a single driver package installs on both 32-bit and 64-bit platforms, it must contain the following:

* Both 32-bit and 64-bit compilations of all kernel-mode drivers, device installation applications, device installers, class installers, and co-installers.
* Decorated INF sections to control platform-specific installation behavior.

In some cases, it might be preferable to provide separate driver packages for 32-bit and 64-bit platforms. We recommend to always decorate your sections with platform-specific extensions when possible. That means, SourceDisksFiles and SourceDisksNames, DDInstall, and Models sections should be decorated whenever possible. This avoids accidental installation of platform-incompatible INFs by uninformed users.

For example, mydriver.inf targets booths x86 and AMD64. A common way to layout this package is the following:

mydriver.inf (with decorated SourceDisksFiles, SourceDisksNames, …)

mydriver.cat

AMD64

 mydriver.sys

X86

 mydriver.sys

## DPInst Installation Packages

A platform-specific version of DPInst must be provided for each target platform that the INF supports.

Single driver package for 32-bit and 64-bit platforms. All files in the driver package must reside in a single directory with both versions of DPInst.exe, which must be renamed. \*install.exe is the application, which will be discussed later in this paper. For example:

Driver32and64\
 Install.exe \* (launches either DPInst32.exe or DPInst64.exe)
 DPInst32.exe (32-bit DPInst.exe renamed to DPInst32.exe)
 DPInst64.exe (64-bit DPInst.exe renamed to DPInst64.exe)

 mydriver.inf

 AMD64

 mydriver.sys (to be installed on 64-bit platform)

 X86

 mydriver.sys (to be installed on 32-bit platform)

 …

For more advanced options with DPInst, see “Using the Driver Package Installer (DPInst).” DPInst can be configured to search for drivers in specific subdirectories, in which case they are not required to reside in the same directory as the INF.

Separate driver packages for both 32-bit and 64-bit platforms. Each driver package must reside in a separate directory together with the version of DPInst.exe for the target platform. For example:

Install.exe

Driver32\
 Driver32.inf
 DPInst.exe (32-bit version)
 …

Driver64\
 Driver64.inf
 DPInst.exe (64-bit version)
 …

In this example, Install.exe is an application, which will be discussed later in this paper.

## Driver Package Installers

To be capable of installing either 32-bit or 64-bit driver packages, a 32-bit installer must be able to detect the platform on which it is running. On 64-bit platforms, the 32-bit installer runs under WOW64.

An installer—Installer.exe—can detect the platform on which it is running in any of these ways:

* Call IsWow64Process with a handle to your process and a pointer to a Boolean value.

If the process is running under WOW64, IsWow64Process sets the Boolean value to TRUE; otherwise, it sets the Boolean value to FALSE.

* Check the value returned by UpdateDriverForPlugAndPlayDevices. If the return value is ERROR\_IN\_WOW64, the 32-bit application is executing on a 64-bit platform.
* Call GetSystemWow64Directory.

If GetSystemWow64Directory fails and GetLastError returns ERROR\_CALL\_NOT\_IMPLEMENTED, the installer is running on 32-bit Windows.

The installer can then call CreateProcess to launch DPInst for the target platform:

* For 32-bit platforms, the installer launches the 32-bit version of DPInst, which installs the 32-bit driver package.
* For 64-bit platforms, the installer launches the 64-bit version of DPInst, which installs the 64-bit driver package.

## MSI Installation Packages with DIFxApp

DIFxApp can be used to install MSI installation packages for drivers that are associated with user-mode applications. Each MSI package must include the DIFxApp.msm merge module for the target platform.

For driver packages that are associated with applications, a Microsoft Windows Installer (MSI) package must be created for each platform.

# Examples

The examples in this section show how to launch the correct version of DPInst for 32‑bit Windows or 64-bit Windows.

## Launching the Correct Version of DPInst from an Installer

The following code example shows how an installer can launch the correct version of DPInst for 32-bit Windows or 64-bit Windows. Briefly, this example does the following:

1. Determines whether the installer is running on 32-bit Windows or 64-bit Windows.

2. Launches Dpinstx86.exe for 32-bit Windows or Dpinstx64.exe for 64-bit Windows.

**Determine whether the installer is running on 32-bit Windows or 64-bit Windows.** The following fragment initializes variables for the CreateProcess call and calls a helper function—IsWow64 —to determine whether the installer is running on 32-bit Windows or 64-bit Windows. The code for IsWow64 is shown later in this paper.

IsWow64 returns TRUE if the installer is running on 64-bit Windows and FALSE if the installer is running on 32-bit Windows. On the basis of this return value, the code fragment assigns “dpinstx86.exe” or ”dpinstx64.exe” to a variable named AppName and then assigns the value of AppName to a variable named command:

|  |
| --- |
| #include <stdio.h>#include <windows.h>#include <tchar.h>#include <strsafe.h>static BOOL IsWow64( void );static void PrintLastError( const WCHAR\* apiname );int \_\_cdecl wmain( int argc, wchar\_t\* argv[] ){ LAUNCHUPINFO LaunchupInfo; PROCESS\_INFORMATION ProcessInformation; UNREFERENCED\_PARAMETER(argc); UNREFERENCED\_PARAMETER(argv); wprintf( L"Running dpinst depending on the underlying platform\n" ); WCHAR \* X86 = TEXT("dpinstx86.exe");// For separate driver package, WCHAR \* X86 = TEXT(“Driver32\dpinst.exe”) WCHAR \* X64 = TEXT("dpinstx64.exe");// For separate driver package, WCHAR \* X64 = TEXT(“Driver64\dpinst.exe”) WCHAR \* AppName = NULL; WCHAR \* command = NULL; if (IsWow64())  { AppName = X64; wprintf( L"We are on a 64 bit OS.\n" ); } else { AppName = X86; wprintf( L"We are on a 32 bit OS.\n" ); } command = (WCHAR\*)HeapAlloc( GetProcessHeap(), 0, (wcslen(AppName)+1)\*sizeof(WCHAR) ); if( command == NULL ) { wprintf( L"Out of memory.\n" ); return 0; }  StringCchCopy( command, wcslen(AppName)+1, AppName ); ZeroMemory( &ProcessInformation, sizeof( ProcessInformation ) ); ZeroMemory( &LaunchupInfo, sizeof( LaunchupInfo ) ); LaunchupInfo.cb = sizeof( LAUNCHUPINFO );   |

**Call CreateProcess to launch the correct version of DPInst.** The example calls CreateProcess, passing command as the command line to execute (lpCommandLine). If CreateProcess succeeds, it returns a nonzero value and creates a new process and its primary thread.

The new process runs the specified executable file (Dpinstx86.exe or Dpinstx64.exe) in the security context of the calling process. The installer waits for the process to finish (WaitForSingleObject) and then cleans up by closing handles and freeing memory allocated for the process:

|  |
| --- |
| if( CreateProcess(  NULL,  command,  NULL,  NULL,  FALSE,  0,  NULL,  NULL,  &LaunchupInfo,  &ProcessInformation ) )  { wprintf( L"CreateProcess succeeded\n" ); } else { PrintLastError( L"CreateProcess()" );  } if( ProcessInformation.hProcess ) { DWORD Ret = WaitForSingleObject( ProcessInformation.hProcess, INFINITE );  if( WAIT\_OBJECT\_0 == Ret ) wprintf(L"Process finished.\n" ); else PrintLastError( L"WaitForSingleObject"); DWORD ExitCode; if( GetExitCodeProcess( ProcessInformation.hProcess, &ExitCode ) ) wprintf( L"Process exit code = 0x%X\n", ExitCode ); else PrintLastError( L"GetExitCodeProcess()" );  GetExitCodeThread( ProcessInformation.hThread, &ExitCode ); wprintf( L"Thread exit code = 0x%X\n", ExitCode );  CloseHandle( ProcessInformation.hProcess ); CloseHandle( ProcessInformation.hThread ); } else { wprintf( L"No process created\n"); } if( command )  HeapFree( GetProcessHeap(), 0, command );  return 0;} |

**IsWow64.** The IsWow64 function determines whether the installer is running on 32‑bit Windows or 64-bit Windows by calling GetSystemWow64Directory. If this function fails and GetLastError returns ERROR\_CALL\_NOT\_IMPLEMENTED, the installer is running on 32-bit Windows:

|  |
| --- |
| typedef UINT (WINAPI\* GETSYSTEMWOW64DIRECTORY)(LPTSTR, UINT);BOOLIsWow64( void ){#ifdef \_WIN64 return FALSE;#else GETSYSTEMWOW64DIRECTORY getSystemWow64Directory; HMODULE hKernel32; TCHAR Wow64Directory[MAX\_PATH]; hKernel32 = GetModuleHandle(TEXT("kernel32.dll")); if (hKernel32 == NULL) { // // This shouldn't happen, but if we can't get  // kernel32's module handle then assume we are  //on x86. We won't ever install 32-bit drivers // on 64-bit machines, we just want to catch it  // up front to give users a better error message. // return FALSE; } getSystemWow64Directory = (GETSYSTEMWOW64DIRECTORY) GetProcAddress(hKernel32, "GetSystemWow64DirectoryW"); if (getSystemWow64Directory == NULL) { // // This most likely means we are running  // on Windows 2000, which didn't have this API  // and didn't have a 64-bit counterpart. // return FALSE; } if ((getSystemWow64Directory(Wow64Directory, sizeof(Wow64Directory)/sizeof(TCHAR)) == 0) && (GetLastError() == ERROR\_CALL\_NOT\_IMPLEMENTED)) { return FALSE; }  // // GetSystemWow64Directory succeeded  // so we are on a 64-bit OS. // return TRUE;#endif} |

**PrintLastError.** The **PrintLastError** function is part of the error handling code. In the following example, **PrintLastError** prints the error string for the last-error code set by a function:

|  |
| --- |
| voidPrintLastError( const WCHAR\* apiname ){ DWORD error = GetLastError(); LPVOID lpvMessageBuffer; if( FormatMessage(FORMAT\_MESSAGE\_ALLOCATE\_BUFFER | FORMAT\_MESSAGE\_FROM\_SYSTEM, NULL, error,  MAKELANGID(LANG\_NEUTRAL, SUBLANG\_DEFAULT), //The user default language (LPTSTR)&lpvMessageBuffer, 0, NULL) ) { \_tprintf( TEXT("ERROR: %s: 0x%X: %s\n"), apiname, error, lpvMessageBuffer ); LocalFree(lpvMessageBuffer); }} |

## Launching the Correct Version of DPInst from a Command Script

A command script can be used as an alternative to a full-blown installer to install a 32-bit or 64-bit driver package, as shown in the following example:

|  |
| --- |
| @if /I "%PROCESSOR\_ARCHITECTURE%" == "AMD64" (  REM echo Installing driver from 64-bit installer on 64-bit OS  dpinstx64.exe) else if /I "%PROCESSOR\_ARCHITECTURE%" == "X86" (  @if /I "%PROCESSOR\_ARCHITEW6432%" == "AMD64" (    REM echo Installing driver from 32-bit installer on 64-bit OS (WOW64)    dpinstx64.exe //// For separate driver package, x64DIR\dpinst.exe)  ) else (    REM echo Installing driver from 32-bit installer on 32-bit OS    dpinstx86.exe// For separate driver package, x86DIR\dpinst.exe)  )) |

# Resources

If you have questions about installing 64-bit driver packages, send e-mail to difxbeta@microsoft.com.

Driver Install Frameworks Tools

Driver Install Frameworks Overview

<http://www.microsoft.com/whdc/driver/install/difxtools.mspx>

Driver Package Installer (DPInst)

<http://msdn.microsoft.com/en-us/library/ms790308.aspx>

Driver Install Frameworks for Applications (DIFxApp)

<http://msdn.microsoft.com/en-us/library/ms790295.aspx>

Installation

Installation and Driver Signing on WHDC

<http://www.microsoft.com/whdc/driver/install/default.mspx>

Device Installation in the Windows Driver Kit (WDK)

<http://msdn.microsoft.com/en-us/library/ms791091.aspx>

About Windows Installer

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/msi/setup/about_windows_installer.asp>

64-bit Platforms

64-bit System Design on WHDC

<http://www.microsoft.com/whdc/system/platform/64bit/default.mspx>

64-bit Issues in the WDK

<http://msdn.microsoft.com/en-us/library/aa489627.aspx>

   INF Requirements for 64-bit Systems

<http://www.microsoft.com/whdc/driver/install/64INF_reqs.mspx>