

# Plug and Play Device Driver Deployment in Windows Vista and Windows Server Longhorn

---

Version 1.0 – April 10, 2007

---

## **Abstract**

This paper provides information about Plug and Play driver deployment scenarios for the Windows Vista™ and Windows Server® Code Name "Longhorn" operating systems. Original equipment manufacturers (OEMs), corporate IT support staff, and driver developers can use the information in this paper to better understand driver deployment scenarios and debugging techniques for Windows Vista.

This document is intended to supplement information in the Windows OEM Preinstall Kit (OPK) and Windows Automated Installation Kit (Windows AIK). It includes an overview of the Windows Vista and Windows Server Longhorn deployment architecture, driver deployment scenarios, definitions, and common device deployment problems and errors.

This information applies for the following operating systems:

- Windows Server Longhorn

- Windows Vista™

- Microsoft® Windows Server 2003

- Microsoft Windows XP

- Microsoft Windows 2000

The current version of this paper is maintained on the Web at:

[http://www.microsoft.com/whdc/driver/install/PnP\\_drv-deploy.msp](http://www.microsoft.com/whdc/driver/install/PnP_drv-deploy.msp)

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

**Contents**

Introduction .....3  
Overview of Device Driver Deployment in Windows Vista .....3  
Using Package Manager to Add Device Drivers .....3  
Using Unattend.xml to Add Drivers .....5  
    Adding Device Drivers during Windows Setup .....6  
    auditSystem .....6  
Sysprep Generalize with PersistAllDeviceInstalls Set to True.....6  
Using DPInst to Add Device Drivers during Audit User Mode .....7  
Windows Server Longhorn Automatic Driver Injection .....8  
Troubleshooting Driver Deployment: Plug and Play Ranking .....9  
    Class-Specific Ranking Rules .....9  
    INF Syntax .....10  
Walkthrough: Diagnosing Device Driver Deployment Problems .....11  
    Gathering Information .....11  
    Detecting and Querying Hardware IDs and Compatible IDs: SetupAPI.dev.log  
    Walkthrough .....12  
    Searching for Driver Matches: SetupAPI.dev.log Walkthrough .....13  
    Best Matches Based on Driver Ranking: SetupAPI.dev.log Walkthrough .....14  
    PkgMgr Offline Driver Installation Log Format.....15  
TBD: Checklist: Summary Actions for Deploying Drivers .....16  
Glossary.....16  
Resources.....18

**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.

© 2007 Microsoft Corporation. All rights reserved.

Microsoft, Authenticode, Win32, Windows, 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.

## Introduction

---

The Windows Vista™ and the Windows Server® Code Name "Longhorn" operating systems introduce a new componentization, image-based setup (IBS) deployment architecture that changes the way images are deployed from earlier versions of Microsoft® Windows® operating systems. This paper provides information about how Plug and Play device drivers are deployed during setup. This includes architectural details, an overview of common driver deployment scenarios, "tricks of the trade," a preview of Windows Server Longhorn updates, and techniques for troubleshooting device driver deployment. This paper does not supplant how-to information already available in the OEM Preinstall Kit (OPK) and Windows Automated Installation Kit (Windows AIK).

For definitions of terms used in this paper, see "Glossary" at the end of this paper.

## Overview of Device Driver Deployment in Windows Vista

---

The new IBS model introduces two paths for driver deployment: offline management of images and online actions that occur in the context of the operating system. Many of the actions including third-party driver installation that occur during the text-mode phase of Setup on previous Windows operating systems now occur during the Windows preinstallation environment (WinPE) offline phases of Setup where operating system images can be edited.

Device drivers can be deployed before or during setup by using several methods. The best choices vary, depending on the deployment techniques being used. Driver deployment paths include:

- Using Package Manager to add drivers to an offline image.
- Using answer files to add drivers during the offline phase of Setup.
- Using answer files to add drivers during the AuditSystem online phase of Setup.
- Using answer files and the DPInst.exe tool to add drivers online during the AuditUser phase of Setup.
- For Windows Server Longhorn, placing drivers in the \$WinPEDrivers\$ directory to be picked up automatically during operating system setup.

**Note Due** to changes in architecture changes and the deprecation of text-mode Setup, configuring the **DevicePath** key to automatically inject drivers from a given location during Setup is *no longer supported*.

These methods stage drivers to the driver store and, if these drivers are boot critical, install them so that they are ready for boot. However, full device installation can occur only on an online image that is running the operating system. This paper also covers the preparation of images by using Syprep **generalize** and **specialize** in how they relate to driver and device installation.

## Using Package Manager to Add Device Drivers

---

You can use Package Manager (Pkgmgr.exe) to install one or more device drivers to an offline Windows image. This enables you to add boot-critical device drivers to Windows before installation.

When you use Package Manager to install a device driver to an offline image, the device driver is added to the driver store, a location on the computer that contains

all the drivers for that computer. When Plug and Play runs, detected devices are matched with device drivers in the driver store. The driver store replaces the Drivers.cab file in Windows XP and earlier operating systems.

Boot-critical drivers are reflected on the system. Driver reflection is the process of installing a driver on a computer that might or might not have a device for that driver. Typically, this involves copying the driver files to a destination location and creating the service.

Detailed instructions and sample answer files are available in the Windows OPK and Windows AIK documentation under "Add Device Drivers to an Offline Windows Image." Drivers added via this method are still subject to Plug and Play ranking rules described later in this paper. It is also important to note that all drivers in the directory and subdirectories that are referenced in the answer file are added to the image. You must manage these answer files and directories carefully to address concerns about increasing the size of the image with unnecessary driver packages.

One thing to note while using Pkgmgr to add device drivers to the offline image is that you cannot depend on the Pkgmgr return code for verification that the drivers were added successfully to the driver store of the image. Any errors that are encountered are filtered out so that Pkgmgr can go through and attempt adding all drivers that are found on the share or directory listed in the answer file. The recommended way of checking for errors is to use the "/l:" flag passed on the command line to look at the log file generated by Pkgmgr. An example of the log is provided at the end of the paper. This behavior might be made flexible in future releases of Windows.

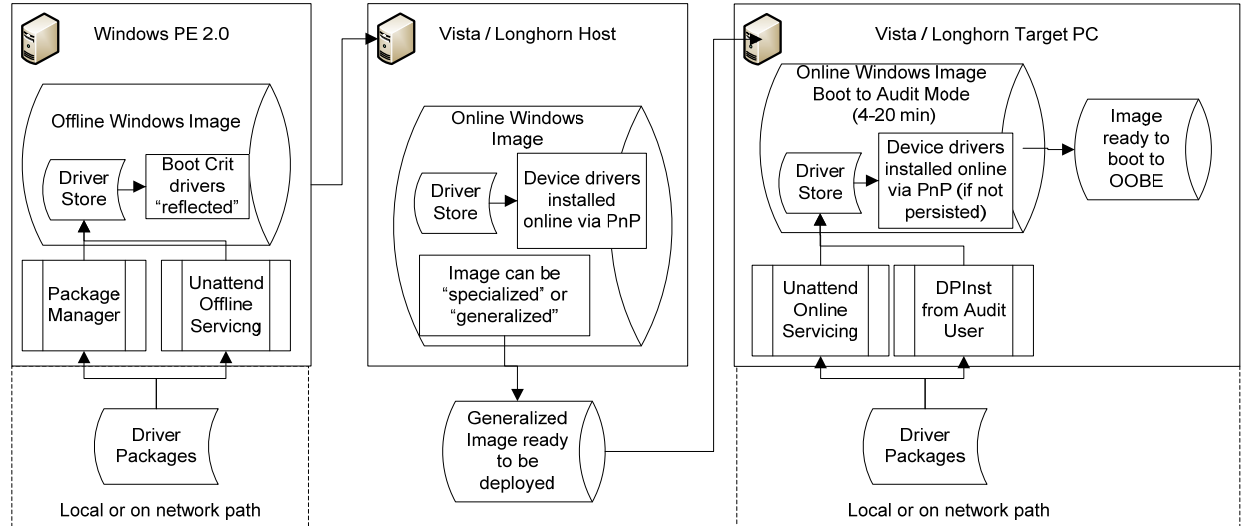
An immediate workaround is to scan the drivers.log file generated by Pkgmgr after the driver -dd operation for the phrase "ERROR" because this phrase exists only in the log file if Pkgmgr encounters an error while adding a driver package to the store. The drivers.log file can be generated by adding the following registry key on the host system:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Device Installer
DebugPkgmgr          DWORD          0x1
```

## Using Unattend.xml to Add Drivers

Configuration passes are phases of Windows Setup that are used to apply settings in an unattended installation answer file.

The following diagram shows the relationship between the configuration passes and the valid configuration passes for different executables.



The following table describes the different configuration passes.

Configuration pass	Description from Windows AIK	Plug and Play device or driver actions
<b>offlineServicing</b>	Applies updates to a Windows image. Also applies packages, including software fixes, language packs, and other security updates.	Drivers can be added offline by using the offlineServicing pass of Unattend. Boot-critical devices are also "reflected" or installed.
<b>specialize</b>	Creates and applies system-specific information. For example, you can configure network settings, international settings, and domain information.	Installed boot-critical drivers not required on the current machine are disabled. "Phantom" device installations for devices not on the current machine are removed.
<b>generalize</b>	Enables you to minimally configure <b>sysprep /generalize</b> , as well as configure other Windows settings that must persist on your reference image. The <b>sysprep /generalize</b> command removes system-specific information. For example, the unique security ID (SID) and other hardware-specific settings are removed from the image. The <b>generalize</b> pass runs only if you run <b>sysprep /generalize</b> .	The boot-critical drivers are reenabled that were disabled during Sysprep <b>specialize</b> so that the image can boot on a variety of hardware. "Phantom" device installations for devices not on the current machine are removed.
<b>auditSystem</b>	Processes unattended Setup settings while Windows is running in system context, before a user logs on to the computer in audit mode. The auditSystem pass runs only if you boot to audit mode.	Drivers can be added by using the online pass of unattend. Drivers are installed if the device is present and the driver is the best Plug and Play ranking match.

Configuration pass	Description from Windows AIK	Plug and Play device or driver actions
<b>auditUser</b>	Processes unattended Setup settings after a user logs on to the computer in audit mode. The auditUser pass runs only if you boot to audit mode.	Scripts can be used to launch the DpInst.exe tool from the Driver Install Frameworks (DIFx) to add drivers. /SH switch can be used only to target present hardware.
<b>oobeSystem</b>	Applies settings to Windows before Windows Welcome starts.	No driver or device actions.

**Note** Not all configuration passes run in a given installation of Windows. Some passes, such as auditSystem and auditUser, run only if you boot to audit mode.

## Adding Device Drivers during Windows Setup

You can install additional device drivers during Windows Setup by creating an answer file. In this answer file, you can specify the paths to device drivers on a network share (or a local path) and the configuration passes in which you intend to install them.

You can install device drivers in the windowsPE, offlineServicing, or auditSystem configuration passes.

By adding device driver paths to the windowsPE or offlineServicing configuration passes, you can add out-of-box device drivers to the Windows image before the system starts. This enables you to add boot-critical device drivers to a Windows image.

Adding drivers to auditSystem enables you to add additional out-of-box drivers during audit mode. This enables you to maintain a simple Windows image and then add only the drivers that are required for a specific hardware configuration, by using a script to call DpInst during the auditSystem phase.

## auditSystem

The auditSystem pass processes unattended Setup settings in system context in audit mode. The auditSystem pass runs immediately before the auditUser pass, which is used to apply settings in user context.

Typically, auditSystem is used to add additional device drivers and assign a name to the system specific for audit mode.

Audit mode enables OEMs and corporations to install additional device drivers, applications, and other updates. When Windows boots to audit mode, the auditSystem and auditUser unattended Windows Setup settings are processed.

By using audit mode, you can maintain fewer Windows images because you can create a reference image with a minimal set of drivers. The image can be updated with additional drivers during audit mode. You can then test and resolve any issues related to malfunctioning or incorrectly installed devices on the Windows image.

## Sysprep Generalize with PersistAllDeviceInstalls Set to True

The PersistAllDeviceInstalls setting is used when running **generalize** on a system identical or similar to the target system. The advantage is that currently installed devices remain installed as long as the device is present on the system. (Both Sysprep **specialize** and **generalize** clean up “phantom” device nodes or device installations for devices not present.)

If the image on which Sysprep has been run is deployed to a system with a different hardware configuration, the system may experience unexpected device removals or configuration changes that can cause instability, so use caution and test carefully when setting to true.

Sample performance tests show a savings of 5 to 11 minutes between first boot and the Windows Welcome screen due to decreased device installation and associated activity.

The following is a sample answer file:

```
<unattend xmlns="urn:schemas-microsoft-com:unattend" xmlns:wcm="http://foo">
  <settings pass="generalize">
    <component name="Microsoft-Windows-PnpSysprep"
publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS"
processorArchitecture="x86">
      <PersistAllDeviceInstalls>true</PersistAllDeviceInstalls>
    </component>
  </settings>
</unattend>
```

## Using DPNst to Add Device Drivers during Audit User Mode

---

The auditUser pass processes unattended Setup settings in user context in audit mode. The auditUser pass runs after the auditSystem pass, which is used to apply settings in system context.

AuditUser is used to execute RunSynchronous or RunAsynchronous commands. These commands are used to run scripts, applications, or other executables during audit mode.

In Windows Vista, when adding drivers to an offline image by using unattend from WinPE, the Plug and Play driver plug-in makes no assumptions about the hardware to which the image will be deployed. In some scenarios, the image is for another system. Also an OEM may want to add third-party drivers for devices or hardware currently not present or discoverable before Windows Server Longhorn.

In some OEM and corporate IT environments, a single large share of drivers is maintained for a variety of systems being deployed. In these scenarios, using unattend or Package Manager to add drivers means that many drivers may be imported into the driver store on a given image when only a fraction of those drivers are required. This can create performance and disk space problems, especially if the driver share is large.

Microsoft recommends that you carefully manage driver shares and unattend scripts so that only required drivers are added to each image. If that is not practical, it is possible to add non-boot-critical drivers online by using the Audit User pass of unattend. (Boot-critical drivers must be installed offline to make it to this point.) If this solution is not used, the redistributable DPNst.exe tool available in the Windows Driver Kit (WDK) can be called from a script and has flags that can be used to selectively add drivers only if the hardware is present.

The following is an excerpt from the DPNst documentation:

```
/sh - Sets the scanHardware flag to ON, which configures DPNst to install a driver package for a Plug and Play (PnP) function driver only if the driver package matches a device that is currently configured in a computer and if the driver package is a better match for the device than the driver package that is currently installed on the device.
```

```

From "RunSynchronous" in Unattend.chm
<RunSynchronous>
  <RunSynchronousCommand wcm:action="add">
    <Credentials>
      <Domain>MyDomain</Domain>
      <Password>MyPassword</Password>
      <Username>MyUsername</Username>
    </Credentials>
    <Description>MySynchCommand1</Description>
    <Order>1</Order>
    <Path>\\network\server\share\filename</Path>
    <WillReboot>OnRequest</WillReboot>
  </RunSynchronousCommand>
  <RunSynchronousCommand wcm:action="add">
    <Credentials>
      <Domain>MyDomain</Domain>
      <Password>MyPassword</Password>
      <Username>MyUsername</Username>
    </Credentials>
    <Description>MySynchCommand2</Description>
    <Order>2</Order>
    <Path>\\network\server\share\filename</Path>
    <WillReboot>OnRequest</WillReboot>
  </RunSynchronousCommand>
</RunSynchronous>

```

**Note** The DIFx tool set might be deprecated and replaced in the future by a new set of APIs.

## Windows Server Longhorn Automatic Driver Injection

In Windows Vista, the driver injection during Setup depends solely on the Unattend answer file during the WinPE phase of Setup. There is no automatic pulling of drivers from a \$OEM\$ directory, as existed in Windows XP, due to changes in the Windows Setup model.

However, because many server systems ship without a preinstalled operating system, Windows Server Longhorn requires support to automatically search for predetermined directories on the system to look for drivers. During WinPE the system looks for the directory named \$WinPEDriver\$ at the root of all visible drivers given a drive letter of C or greater. If this directory exists, the module then adds this path to the list of paths that it maintains to search for driver packages. When this operation is complete, the module continues to scan the answer file, if present, for additional driver paths.

Because of this operation, drives that contain a WinPEDriver\$ directory in the root cause Setup to recursively search this directory for driver packages to be imported into the image during the WinPE phase. This includes hard drive partitions and removable media like floppy disk drives and flash drives. Type-27 hidden partitions are assigned a driver letter during the WinPE phase of Setup and are also searched.

Drive letters A and B are not searched during this operation. These drives are considered to be reserved for floppy disk driver media and, due to potential error conditions, are omitted from the search algorithm.

This enables scenarios in which a server system is shipped without an operating system and allows OEMs more flexibility in provisioning boot storage and other drivers to enable Setup on these systems. One potential consequence is a situation in which WinPE cannot access the hard drive that contains the \$WinPEDriver\$ directory because the storage driver has not yet been loaded in WinPE. This dilemma can be solved in one of the following ways:

- Boot to a customized WinPE image that already has the storage driver installed.
- Place drivers on an embedded flash drive that is accessible from WinPE.
- Place drivers on removable media such as flash or CD Rom. (Floppy disk drives are not supported.)

## **Troubleshooting Driver Deployment: Plug and Play Ranking**

---

One of the most common issues in deploying drivers occurs when a driver is successfully imported into the driver store but, after the system is online, Plug and Play finds a higher ranking driver and installs that driver instead.

The Windows PnP manager ranks the following driver package properties in order of importance:

1. Signing
2. Plug and Play ID match
3. Driver date
4. Driver version

For detailed documentation, see “How Setup Ranks Drivers (Windows Vista)” in the WDK.

For example, if a device has a better Plug and Play ID match but is unsigned, a signed driver with a compatible ID match takes precedence. Likewise, a newer driver can be outranked by an older driver if the older driver has a better Plug and Play ID match or signature.

In Windows Vista, there is a difference between drivers that are signed by Microsoft—different classes of Windows Hardware Quality Labs (WHQL)-type signatures—and drivers that are signed by Microsoft Authenticode®. The Microsoft WHQL signatures take precedence, and Authenticode-signed drivers outrank unsigned drivers. However, a Premium Logo signature does not outrank another Microsoft signature, and a Windows Vista signature does not automatically outrank drivers signed by Windows XP, except for the device classes listed in the following section.

**Note** Most Windows Vista inbox drivers have a driver date of 6/21/2006.

### **Class-Specific Ranking Rules**

Driver class-specific rules apply in driver ranking, in addition to the Plug and Play base rules. The following classes have a policy to rank some operating system version signatures higher than others. For example, those marked with a lower logo version of 6.0 always favor a Windows Vista-signed driver over other operating system signatures; 5.2 favors Windows Server 2003 or Windows Vista; and 5.1 favors Windows XP, Windows Server 2003, and Windows Vista.

All device classes not listed here have a default operating system logo version of 5.0, which means that any operating system signature from Windows 2000 through Windows Vista is treated equally.

<b>Class name</b>	<b>Lower logo version</b>
1394	5.2 – Windows Server 2003
Display	6.0 – Windows Vista
Image	5.2 – Windows Server 2003
Infrared	5.2 – Windows Server 2003
Media	5.1 – Windows XP
MPS	5.2 – Windows Server 2003
Net	6.0 – Windows Vista
Print	5.2 – Windows Server 2003
Unknown	6.0 – Windows Vista
USB	5.2 – Windows Server 2003
<b>All others</b>	<b>5.0 – Windows 2000</b>

For more information on operating system versioning, see "OSVERSIONINFO" on MSDN.

Display is more complex due to additional rules enforced by the display class installer. The following display driver package properties are shown in order of importance:

1. Windows Vista signed by WHQL or inbox-signed (Premium, Standard, and inbox signatures)
2. Legacy WHQL-signed and Authenticode-signed
3. Unsigned
4. Feature criteria (for example, Windows Vista display driver model [WDDM] drivers are preferred over Windows XP display driver model [XDDM])
5. Plug and Play ID match
6. Driver date
7. Driver version

**Important** The inbox Windows Vista VGA drivers outrank existing XDDM drivers when the user performs an operating system upgrade. This prevents problems with XDDM drivers that have not been validated on Windows Vista. After upgrade, if the user selects **Search for the best available driver on the system** in the Hardware Wizard, the migrated XDDM driver is then made available to the user.

## **INF Syntax**

Another common mistake is to attempt to install a device driver that is not intended for this architecture or operating system. INF syntax such as AMD64 decorations or OSVersion sections should be checked. As a general rule, you should attempt to install any problem package online before further troubleshooting.

## Walkthrough: Diagnosing Device Driver Deployment Problems

---

This section describes the basic steps for gathering information and then using the SetupAPI logs to begin troubleshooting a device installation problem.

### Gathering Information

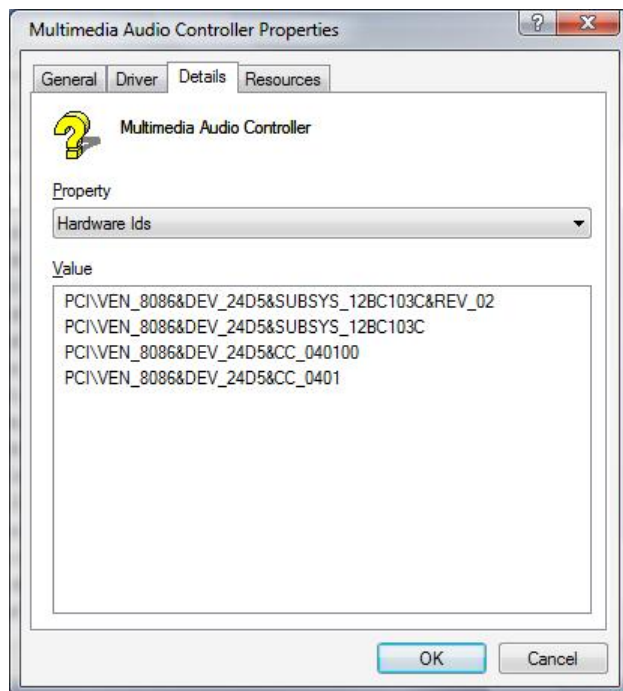
If you are investigating a device installation problem, collect the following information:

- Driver package, including the INF and all files  
These might be located in the driver store repository. For example, the inbox Bluetooth driver package resides in:  
%windir%\System32\DriverStore\FileRepository\bth.inf\_a02df405\  
The numbers after the *.inf* name are generated automatically to create a strong name when imported into the driver store.
- SetupAPI.dev.log and SetupAPI.app.log from %windir%\inf\setupapi\*
- SetupACT.log from the %windir%\panther directory  
The log might be needed if this was a driver migration, F6, Package Manager, or Windows Setup-related installation.

We also recommend noting the following information:

- Any error messages and where they occurred
- Any Device Manager (Devmgmt.msc) device error codes (Code 28, Code 10, and so on)  
Windows Vista error codes are the same as those in Windows XP.  
For a list of Device Manager error codes and suggested solutions, see “Device Manager Error Messages” in the WDK.

- The Hardware ID information for the device  
As shown in the following figure, in the device's **Properties** dialog box, click the **Details** tab and select **Hardware IDs** in the **Property** list. Use CTRL+C to copy these.



Notice that Plug and Play installs a driver on a device automatically only if there is a HWID match with an entry in the INF file.

If a device driver package failed to be copied to the driver store, it will not be listed in Device Manager because a device installation was not attempted. To investigate driver store import errors, see "How to Use the SetupACT and SetupAPI Logs" later in this paper.

**Note** Device Installation errors occur for several reasons. It is not the goal of this paper to detail every error and possible cause. When troubleshooting an error and collecting this information, search for known issues by using online resources such as Microsoft Help and Support.

### Detecting and Querying Hardware IDs and Compatible IDs: SetupAPI.dev.log Walkthrough

The following is an example of what happens when a Windows XP-signed display package named Xpgraphics.inf deployed earlier during Setup is compared to the inbox Display.inf VGA driver. In the following section from %windir%\inf\setupapi.dev.log, Setup detects the hardware and queries device hardware IDs and compatible IDs:

```
>>> [Setup online Device Install (Hardware initiated) -
PCI\VEN_8086&DEV_2562&SUBSYS_01261028&REV_01\3&172e68dd&0&10]
>>> Section start 2006/12/08 13:48:31.750
    ump: Creating Install Process: DrvInst.exe 13:48:31.750
    ndv: Retrieving device info...
    ndv: Setting device parameters...
    ndv: Building driver list...
    dvi: {Build Driver List} 13:48:31.796
    dvi: Searching for hardware ID(s):
```

```

dvi:      pci\ven_8086&dev_2562&subsys_01261028&rev_01
dvi:      pci\ven_8086&dev_2562&subsys_01261028
dvi:      pci\ven_8086&dev_2562&cc_030000
dvi:      pci\ven_8086&dev_2562&cc_0300
dvi:      Searching for compatible ID(s):
dvi:      pci\ven_8086&dev_2562&rev_01
dvi:      pci\ven_8086&dev_2562
dvi:      pci\ven_8086&cc_030000
dvi:      pci\ven_8086&cc_0300
dvi:      pci\ven_8086
dvi:      pci\cc_030000
dvi:      pci\cc_0300

```

## Searching for Driver Matches: SetupAPI.dev.log Walkthrough

Next, Setup searches for potential driver matches, finding both Display.inf and Xpgraphics.inf in the following example:

```

dvi:      Enumerating INFs from path list 'C:\Windows\INF'
inf:      Opened INF:
'C:\Windows\System32\DriverStore\FileRepository\display.inf_30c9fefaf\display
.inf' ([strings] <src=drvstore>)
inf:      Saved PNF:
'C:\Windows\System32\DriverStore\FileRepository\display.inf_30c9fefaf\display
.PNF' (Language = 0409)
dvi:      Created Driver Node:
dvi:      HardwareID - PCI\CC_0300
dvi:      InfName -
C:\Windows\System32\DriverStore\FileRepository\display.inf_30c9fefaf\display
.inf
dvi:      DevDesc - Standard VGA Graphics Adapter
dvi:      DrvDesc - Standard VGA Graphics Adapter
dvi:      Provider - Microsoft
dvi:      Mfg - (Standard display types)
dvi:      ModelsSec - Std.Mfg.NTx86
dvi:      InstallSec - vga
dvi:      ActualSec - vga
dvi:      Rank - 0x0dfe2006
dvi:      Signer - microsoft windows
dvi:      Signer Score - INBOX
dvi:      DrvDate - 06/21/2006
dvi:      Version - 6.0.6000.16386
inf:      Opened INF:
'C:\Windows\System32\DriverStore\FileRepository\xpgraphics.inf_ffff0f2a\xpgr
aphics.inf' ([strings] <src=drvstore>)
inf:      Saved PNF:
'C:\Windows\System32\DriverStore\FileRepository\xpgraphics.inf_ffff0f2a\xpgr
aphics.PNF' (Language = 0409)
sig:      {_VERIFY_FILE_SIGNATURE} 13:48:31.953
sig:      Key = xpgraphics.inf
sig:      FilePath =
C:\Windows\System32\DriverStore\FileRepository\xpgraphics.inf_ffff0f2a\xpgra
phics.inf
sig:      Catalog =
C:\Windows\System32\DriverStore\FileRepository\xpgraphics.inf_ffff0f2a\xpgra
phics.cat
! sig:      Verifying file against specific (valid) catalog failed!
(0xe000244)
! sig:      Error 0xe000244: The software was tested for compliance
with Windows Logo requirements on a different version of Windows, and may
not be compatible with this version.
sig:      {_VERIFY_FILE_SIGNATURE exit(0xe000244)} 13:48:32.265
sig:      {_VERIFY_FILE_SIGNATURE} 13:48:32.265
sig:      Key = xpgraphics.inf
sig:      FilePath =
C:\Windows\System32\DriverStore\FileRepository\xpgraphics.inf_ffff0f2a\xpgra
phics.inf
sig:      Catalog =
C:\Windows\System32\DriverStore\FileRepository\xpgraphics.inf_ffff0f2a\xpgra
phics.cat

```

```

sig:          Success: File is signed in catalog.
sig:          {_VERIFY_FILE_SIGNATURE exit(0x00000000)} 13:48:32.281
dvi:          Created Driver Node:
dvi:          HardwareID - PCI\VEN_8086&DEV_2562
dvi:          InfName -
C:\Windows\System32\DriverStore\FileRepository\xpgraphics.inf_ffff0f2a\xpgraphics.inf
dvi:          DevDesc - Graphics(R) 12345G/GL/GE/PE/GV Graphics
Controller
dvi:          DrvDesc - Graphics(R) 12345G/GL/GE/PE/GV Graphics
Controller
dvi:          Provider - Graphics Corporation
dvi:          Mfg - Graphics Corporation
dvi:          ModelsSec - Graphics.Mfg
dvi:          InstallSec - i845G
dvi:          ActualSec - i845G
dvi:          Rank - 0x0fff2001
dvi:          Signer - Microsoft Windows Hardware Compatibility
Publisher
dvi:          Signer Score - Authenticode
dvi:          DrvDate - 06/21/2005
dvi:          Version - 6.14.10.4342
inf:          Searched 2 potential matches in published INF directory
inf:          Searched 27 INFs in directory: 'C:\Windows\INF'
dvi:          {Build Driver List - exit(0x00000000)} 13:48:32.468

```

## Best Matches Based on Driver Ranking: SetupAPI.dev.log Walkthrough

Now that the driver list has been built, Setup selects the best match based on the driver ranking. In the following example, the inbox Display.inf driver outranks the provisioned Xpgraphics.inf driver:

```

ndv: Selecting best match...
dvi: {DIF_SELECTBESTCOMPATDRV} 13:48:32.468
dvi: Using exported function 'DisplayClassInstaller' in module
'C:\Windows\system32\DispCI.dll'.
dvi: Class installer == DispCI.dll,DisplayClassInstaller
dvi: No CoInstallers found
dvi: Class installer: Enter 13:48:32.484
dvi: Class installer: Exit
dvi: Default installer: Enter 13:48:32.656
dvi: {Select Best Driver}
dvi: Selected driver installs from section [vga] in
'c:\windows\system32\driverstore\filerepository\display.inf_30c9fefa\display
.inf'.
dvi: Class GUID of device changed to: {4d36e968-e325-
11ce-bfc1-08002be10318}.
dvi: {DIF_DESTROYPRIVATEDATA} 13:48:32.671
dvi: Class installer: Enter 13:48:32.671
dvi: Class installer: Exit
dvi: Default installer: Enter 13:48:32.796
dvi: Default installer: Exit
dvi: {DIF_DESTROYPRIVATEDATA - exit(0xe000020e)}
13:48:32.796
dvi: Set selected driver complete.
dvi: Selected:
dvi: Description - [Standard VGA Graphics Adapter]
dvi: InfFile -
[c:\windows\system32\driverstore\filerepository\display.inf_30c9fefa\display
.inf]
dvi: Section - [vga]
dvi: Signer - [microsoft windows]
dvi: Rank - [0x0dfe2006]
dvi: {Select Best Driver - exit(0x00000000)}

```

## PkgMgr Offline Driver Installation Log Format

When driver packages are added to an offline image, PkgMgr logs the actions in a text file that can be generated if the '/l:' flag is used when executing PkgMgr. In addition to this, a "driver operations only" log file can be generated by setting a registry key on the host system. This technique is described in "Using Package Manager to Add Device Drivers to an Offline Windows Image" earlier in this paper.

The log file generated has the following structure and format:

```

INFO: ENTER InstallDriversOfflineINFO: Parameter Offline Windows
Directory Path: 'c:\image\windows'INFO: Parameter Offline system drive on
reboot: 'D:'INFO: Offline Image has 'PROCESSOR_ARCHITECTURE_INTEL'INFO:
User has Administrator privileges.INFO: Offline SOFTWARE hive located at:
c:\image\windows\system32\config\SOFTWAREINFO: Loaded hive key name
{bfla281b-ad7b-4476-ac95-f47682990ce7}c:/image/windows/system32/config/SOFTWAREINFO: Found key
'1'INFO: Processing '1'...INFO: Path =
'\\drivershare\pub\drivers'WARNING:No credentials providedINFO: Offline
SYSTEM hive located at: c:\image\windows\system32\config\SYSTEMINFO:
Loaded hive key name {bfla281b-ad7b-4476-ac95-f47682990ce7}c:/image/windows/system32/config/SYSTEMINFO: Searching for
drivers at '\\drivershare\pub\drivers'...INFO: Locating INFs in
'\\drivershare\pub\drivers'INFO: Found file fragment.xmlINFO: Found
directory pnpcomplex-ProgramFiles under search pathINFO: Found directory
AMD64 under search pathINFO: Found file pnpcomplex.dllINFO: Found file
pnpcomplex.sysINFO: Found directory IA64 under search pathINFO: Found
file pnpcomplex.dllINFO: Found file pnpcomplex.sysINFO: Found file
MySample.txtINFO: Found file pnpcomplex.dllINFO: Found file
pnpcomplex.infINFO: Found file pnpcomplex.PNFINFO: Found file
pnpcomplex.sysINFO: Found directory ToasterFunctionDriver under search
pathINFO: Found directory AMD64 under search pathINFO: Found file
toaster.sysINFO: Found file tostrcls.dllINFO: Found file
tostrcol.dllINFO: Found file delta.CATINFO: Found directory IA64 under
search pathINFO: Found file toaster.sysINFO: Found file
tostrcls.dllINFO: Found file tostrcol.dllINFO: Found file
toastco.infINFO: Found file toastco.PNFINFO: Found directory X86 under
search pathINFO: Found file toaster.sysINFO: Found file
tostrcls.dllINFO: Found file tostrcol.dllINFO: Found file
unattend.xmlINFO: Found directory x86 under search pathINFO: Found file
b57nd60x.sysINFO: Found file netb57vx.infINFO: Found file
netb57vxx86.catINFO: Found 3 driver package(s) at
'\\drivershare\pub\drivers'.INFO: Driver Package
'\\drivershare\pub\drivers\pnpcomplex-ProgramFiles\pnpcomplex.inf'INFO:
Driver Package
'\\drivershare\pub\drivers\ToasterFunctionDriver\toastco.inf'INFO:
Driver Package '\\drivershare\pub\drivers\x86\netb57vx.inf'INFO:
Installing driver package '\\drivershare\pub\drivers\pnpcomplex-
ProgramFiles\pnpcomplex.inf' to the currently offline OS ...INFO: The
class GUID for INF '\\drivershare\pub\drivers\pnpcomplex-
ProgramFiles\pnpcomplex.inf' is '4d36e96f-e325-11ce-bfc1-08002be10318'!INFO:
The class GUID '4d36e96f-e325-11ce-bfc1-08002be10318' is boot-
critical.INFO: '\\drivershare\pub\drivers\pnpcomplex-
ProgramFiles\pnpcomplex.inf' is a boot critical driver.SUCCESS:Added
'\\drivershare\pub\drivers\pnpcomplex-ProgramFiles\pnpcomplex.inf' to
offline driver store at
'c:\image\windows\system32\DriverStore\FileRepository\pnpcomplex.inf_1cf680b
f\pnpcomplex.inf'.SUCCESS:Added driver
'\\drivershare\pub\drivers\pnpcomplex-ProgramFiles\pnpcomplex.inf' to the
offline Windows image at
'c:\image\windows\system32\DriverStore\FileRepository\pnpcomplex.inf_1cf680b
f\pnpcomplex.inf'.SUCCESS:Driver package
'\\drivershare\pub\drivers\pnpcomplex-ProgramFiles\pnpcomplex.inf'
installed!SUCCESS:Successfully marked devices for reinstall!INFO:
Installing driver package
'\\drivershare\pub\drivers\ToasterFunctionDriver\toastco.inf' to the
currently offline OS ...INFO: The class GUID for INF
'\\drivershare\pub\drivers\ToasterFunctionDriver\toastco.inf' is '4d36e96f-
e325-11ce-bfc1-08002be10318'!INFO: The class GUID '4d36e96f-e325-11ce-
bfc1-08002be10318' is boot-critical.INFO:

```

```
'\\drivershare\pub\drivers\ToasterFunctionDriver\toastco.inf' is a boot
critical driver.SUCCESS:Added
'\\drivershare\pub\drivers\ToasterFunctionDriver\toastco.inf' to offline
driver store at
'c:\image\windows\system32\DriverStore\FileRepository\toastco.inf_c605b608\t
oastco.inf'.SUCCESS:Added driver
'\\drivershare\pub\drivers\ToasterFunctionDriver\toastco.inf' to the offline
Windows image at
'c:\image\windows\system32\DriverStore\FileRepository\toastco.inf_c605b608\t
oastco.inf'.SUCCESS:Driver package
'\\drivershare\pub\drivers\ToasterFunctionDriver\toastco.inf'
installed!SUCCESS:Successfully marked devices for reinstall!INFO:
Installing driver package '\\drivershare\pub\drivers\x86\netb57vx.inf' to
the currently offline OS ..INFO: The class GUID for INF
'\\drivershare\pub\drivers\x86\netb57vx.inf' is '4d36e972-e325-11ce-bfc1-
08002be10318'!INFO: The class GUID '4d36e972-e325-11ce-bfc1-08002be10318'
is not boot-critical.INFO: '\\drivershare\pub\drivers\x86\netb57vx.inf'
is not a boot critical driver.SUCCESS:Added
'\\drivershare\pub\drivers\x86\netb57vx.inf' to offline driver store at
'c:\image\windows\system32\DriverStore\FileRepository\netb57vx.inf_064354a8\
netb57vx.inf'.SUCCESS:Added driver
'\\drivershare\pub\drivers\x86\netb57vx.inf' to the offline Windows image at
'c:\image\windows\system32\DriverStore\FileRepository\netb57vx.inf_064354a8\
netb57vx.inf'.SUCCESS:Driver package
'\\drivershare\pub\drivers\x86\netb57vx.inf' installed!SUCCESS:Successfully
marked devices for reinstall!INFO: Installed '3' drivers.INFO: RETURN
InstallDriversOffline (0)
```

## TBD: Checklist: Summary Actions for Deploying Drivers

---

To be provided in a future draft.

## Glossary

---

### answer file

An XML file that contains the settings and configurations to apply to a Windows image during installation. The answer file for Windows Setup is commonly called Unattend.xml. You can create and modify this answer file by using Windows System Image Manager (Windows SIM) or the CPI APIs.

### audit mode

A stage of Windows Setup that enables additional customization and testing before deployment. Audit mode replaces factory mode in Windows XP.

### boot-critical driver

A driver that must be available for the operating system to successfully complete the boot process. Boot-critical drivers include all boot-start drivers.

### configuration pass

A phase of Windows installation. Different parts of the Windows operating system are installed in different configuration passes. You can specify Windows unattended installation settings to be applied in one or more configuration passes.

### device driver

A function driver for a hardware device that has a Plug and Play identifier.

### device management and installation

Discovery and installation of available devices, storage and retrieval of device properties, and storage and retrieval of device settings. Only supported mechanisms should be used to perform these tasks.

**driver package**

A collection of all of the files that are required to successfully load the driver. This includes the device information (.inf) file, the catalog file, and all of the binaries that are copied by the .inf file, including the class installer DLL, co-installer DLL, function driver, branding icons or bitmaps, and property page provider DLL.

**image-based setup (IBS)**

The mechanism used to install, deploy, and test the installation image.

**imaging**

The process of capturing an installation of Windows for deployment to one or more destination computers.

**in-box driver**

A driver in the default Windows installation.

**INF file**

A text system information file used for installing Plug and Play device drivers. A Plug and Play INF file installs on a Plug and Play hardware Identification string (HWID) and installs at least one SYS or DLL file.

**OEM Preinstallation Kit (OPK)**

A set of tools, documentation, and samples that enable OEMs to preinstall Windows efficiently on new computers according to the terms of the OEM Licensing Agreement.

**offline installation**

The process of installing packages and other updates to a Windows image that is not currently running. For example, you can update a Windows image with security updates, language packs, or other packages, by using Package Manager.

**offline Windows image**

A Windows image that is not currently running, either an image in a .wim file or a Windows installation on a separate partition.

**online installation**

An installation that requires the Windows image to boot and start Windows Setup. Windows components are configured as Windows Setup initializes the system.

**online Windows installation**

A Windows installation currently running on the computer.

**original equipment manufacturer (OEM)**

A company that typically purchases computer components from other manufacturers, uses the components to build a personal computer, preinstalls Windows onto that computer, and then sells the computer to the public.

**out-of-box driver**

Any device driver that is not included in the default Windows installation.

**Package Manager**

A tool that installs, uninstalls, configures, and updates features and packages for Windows Vista. Called by Windows Setup during a standard installation or update, Package Manager can also be initiated from the command line to install or update Windows Vista feature packages on an offline destination computer, by using an unattended installation answer file.

### **Reflection**

The operation in which the driver files are copied to the appropriate location, registry settings required by the driver are applied, and a service entry is created to load the driver as necessary; also called *driver reflection*. Reflection allows Plug and Play to be able to locate the drivers without having to go through a device installation first and is used in conjunction with adding boot-critical drivers to the system.

### **synchronous**

A scenario in which each application or command runs in the order listed and each item must finish before the next command is run.

### **system context**

The state of Windows after it first starts when some system services are loaded but no user is logged in and per-user settings are not available.

### **System Preparation (Sysprep) tool**

The tool that prepares an operating system for imaging. Sysprep removes system-specific settings and other data that should not be copied to a destination computer. Sysprep also resets the Windows installation to start Windows Welcome or an audit mode.

### **Unattend.xml**

The generic name for the Windows Vista Setup answer file. Unattend.xml replaces all of the answer files in earlier versions of Windows, including Unattend.txt, Winbom.ini, and others.

### **user context**

The state of Windows after a user is logged on and the appropriate account permissions for that account are effective.

### **Windows Hardware Quality Labs (WHQL)**

A Microsoft hardware-testing organization that produces and supports the Microsoft Hardware Compatibility Test Kit for current Microsoft operating systems. Both hardware and software are tested before rights to use the logo are granted.

### **Windows image**

A Windows image (.wim) file that contains one or more Windows images. Language packs can be installed into a .wim file by using the Package Manager tool.

### **Windows Logo Program**

A certification program to help customers identify systems, hardware, and software that meet a baseline definition of platform features and quality goals and ensure a good user experience of Windows.

### **Windows Preinstallation Environment (WinPE)**

A minimal Microsoft Win32® installation environment with limited services. Built on the Windows kernel, WinPE provides an environment to prepare a computer for Windows installation, copies disk images from a network file server, and starts Windows Setup.

## **Resources**

---

### **Windows Hardware Developer Central:**

#### **Driver Installation page**

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

Includes presentations from WinHEC technical sessions

#### **Driver Signing Requirements for Windows (References and White Papers)**

<http://www.microsoft.com/whdc/winlogo/drvsign/drvsign.mspx>

**Plug and Play Device Driver Migration in Windows Vista**

<http://www.microsoft.com/whdc/driver/install/Pnpmigration.mspix>

**Windows Driver Kit (WDK):**

It is important to use DPInst Version 2.1 available only in the Windows Vista WDK, which includes improvements over Version 2.01 that are specific to Windows Vista.

<http://www.microsoft.com/whdc/devtools/wdk/betawdk.mspix>

**Microsoft Developer Network:**

**WDK documentation topics**

Using Driver Install Frameworks (DIFx)

<http://go.microsoft.com/fwlink/?LinkId=87614>

Using Device Installation Functions

<http://go.microsoft.com/fwlink/?LinkId=87615>

Device Manager Error Messages

<http://go.microsoft.com/fwlink/?LinkId=87616>

How Setup Ranks Drivers (Windows Vista)

<http://go.microsoft.com/fwlink/?LinkId=87617>

Troubleshooting Device Installation

<http://go.microsoft.com/fwlink/?LinkId=87618>

**OSVERSIONINFO** (information on operating system versioning)

<http://go.microsoft.com/fwlink/?LinkId=87619>

**Knowledge Base:**

**How Windows determines the most suitable device driver to install during Setup**

<http://support.microsoft.com/default.aspx?scid=kb;en-us;279112>

**TechNet:**

**Windows Automated Installation Kit (Windows AIK)**

<http://technet2.microsoft.com/WindowsVista/en/library/129a1712-e3d8-46c1-bc09-a14349dc67db1033.mspix>

**Newsgroups:**

microsoft.public.development.device.drivers