Best Practices for Developing Printer Drivers

November 5, 2008

Abstract

This paper provides information about printer drivers for the Windows® family of operating systems. It provides printer driver developers and testers with important information about how to build a high-quality printer driver. In particular, it calls out many of the best practices to implement during driver development and testing.

This information applies for the following operating systems:  
 Windows 7

Windows Server® 2008  
 Windows Vista®

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

For the latest information, see:   
 <http://www.microsoft.com/whdc/device/printers/Print_Drv-Dev.mspx>

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.

© 2008 Microsoft Corporation. All rights reserved.

Microsoft, 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.

Document History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Change |  |  |  |
| November 5, 2008 | First publication | | | |

Contents

[Overview 4](#_Toc213054450)

[Setup Guidelines 4](#_Toc213054451)

[INF Best Practices 4](#_Toc213054452)

[Handling Multiple Drivers in a Single Package 4](#_Toc213054453)

[Listing Models for Correct Plug and Play Ranking 4](#_Toc213054454)

[Launching Application Installations during Driver Installation 5](#_Toc213054455)

[Installer Best Practices 5](#_Toc213054456)

[Co-installer Best Practices 5](#_Toc213054457)

[Multifunction Device Guidelines 6](#_Toc213054458)

[Driver Distribution 6](#_Toc213054459)

[Driver Signing 6](#_Toc213054460)

[Windows Update 6](#_Toc213054461)

[Inbox 7](#_Toc213054462)

[Device Configuration 7](#_Toc213054463)

[Identical Ports that Expose the Same Functions 7](#_Toc213054464)

[Devices that Enumerate All Functions 7](#_Toc213054465)

[Autoconfiguration of Installable Options 7](#_Toc213054466)

[Continuing Operation after Device Errors Are Cleared 8](#_Toc213054467)

[Driver Guidelines 8](#_Toc213054468)

[Driver Components that Operate Only in User Mode 8](#_Toc213054469)

[Driver Components that Are Built in the Same Processor Architecture 8](#_Toc213054470)

[Drivers Used Concurrently or from Multiple Processes 8](#_Toc213054471)

[Driver Isolation 9](#_Toc213054472)

[Process Isolation 9](#_Toc213054473)

[Driver Installations that Do Not Prevent Other Drivers from Functioning Correctly 9](#_Toc213054474)

[Unsupported Escapes and Events 9](#_Toc213054475)

[Port Monitors 10](#_Toc213054476)

[Using Microsoft Port Monitors 10](#_Toc213054477)

[Core Drivers 10](#_Toc213054478)

[Using Microsoft Core Drivers 10](#_Toc213054479)

[User Interfaces 11](#_Toc213054480)

[Using Third-Party Core Drivers 11](#_Toc213054481)

[Cluster Scenarios 11](#_Toc213054482)

[Backward and Forward Compatibility 12](#_Toc213054483)

[Security Guidelines 12](#_Toc213054484)

[Working with Limited User Accounts 12](#_Toc213054485)

[Data Execution Prevention 12](#_Toc213054486)

[Responsible Use of Personally Identifiable Information 13](#_Toc213054487)

[Reliability on Secure Printers 13](#_Toc213054488)

[Development and Testing Guidelines 13](#_Toc213054489)

[Development Guidelines 13](#_Toc213054490)

[Windows Driver Kit 13](#_Toc213054491)

[Testing Guidelines 14](#_Toc213054492)

[Print Verifier 14](#_Toc213054493)

[Windows Logo Kit 14](#_Toc213054494)

[Using other Microsoft-Provided Test Tools 15](#_Toc213054495)

[Resources 15](#_Toc213054496)

# Overview

This document provides printer driver developers and testers with important information about how to build a high-quality printer driver. In particular, it describes best practices to follow and other practices to avoid during driver development.

# Setup Guidelines

Setup is the first experience that users have with your device and driver. Setup has three major aspects: INF-based installation, use of installers, and driver distribution.

## INF Best Practices

Correctly constructed INFs are key to ensuring correct setup. We do not recommend installing printer driver components outside an INF because problems with migration or point-and-print are likely to occur.

### Handling Multiple Drivers in a Single Package

You must use the **CopyINF** directive or the **SetupCopyOEMInf** API to copy multiple INF files. This is especially relevant when you try to load multiple drivers for multifunction printer (MFP) support or when you must update core drivers. The **CopyINF** directive is preprocessed and therefore guarantees that the required drivers are all available in the driver store when installation begins. In scenarios with core drivers if you do not use **CopyINF** and the required core driver section is not available, Setup cannot complete the installation.

For more information on writing INFs to take advantage of updated core drivers, refer to “Using Updated Core Print Drivers” in the Windows Driver Kit (WDK).

### Listing Models for Correct Plug and Play Ranking

Plug and Play evaluates the individual lines in the models section of the INF as separate devices. For Plug and Play to function correctly, you must follow several rules:

* Never list the same hardware identifier (HWID) on two different lines in the same models section.
* Never list multiple HWIDs for different devices on the same line in a models section.
* Always order HWIDs in a single line from most specific to least specific (left to right).
* If you have a compatible ID for a printer, never list it in the HWID position (first). Always leave the HWID position blank and place the compatible ID in the second position.

For more information on Plug and Play ranking and formatting the models section of an INF, refer to “INF Models Section” in the WDK.

### Launching Application Installations during Driver Installation

You must implement value-added application installation by using extensions to the Found New Hardware Wizard in the form of finish-install pages. To install all value-added software, open the installer from the INF file that is provided with the driver package.

Except for finish-install pages, do not call an executable programmatically from a device installation. Complying with these best practices improves the ability of a driver to correctly migrate, operate in cluster failover, and support point-and-print.

## Installer Best Practices

Installers should never be required to make a device function correctly. This would limit the ability of the driver to be distributed on Windows Update, to be included on the OS media, or to install properly through Plug and Play. If you decide an installer is required for a better user experience outside of these cases, the following is some guidance on creating an installer that works well with Windows:

* Do not set a maximum version check in the installer. This action means that the installer will not be compatible with later versions of Windows even if the application and driver can work on the new operating system. As a result, customers will complain and there may be engineering cost incurred to support the new operating system.
* Distribute the driver files as a signed INF package and require the installer to use that INF to install the driver. If the installer extracts the signed INF package to a temporary directory, users can install the INF package even if the actual installer fails.
* Do not try to circumvent driver signing, the driver store, or other mechanisms that are meant to ensure system stability and security for end users. Microsoft ships security updates to each operating system as flaws are identified, and your driver can stop installing if it takes advantage of a hole that is later closed.
* Do not try to enforce relationships between drivers and applications that the operating system does not recognize. Poorly written drivers might not work correctly if certain applications are not running on the system. Writing an installer to ensure that the required application is in place is not a solution. Users can uninstall applications at any time and, as of the time this document was written, there is no way for the operating system to track dependencies between drivers and applications. You should write drivers that fail gracefully if dependencies are not met instead of trying to work around the problem in an installer.

### Co-installer Best Practices

Co-installers should only be used in exceptional circumstances. Co-installers have several limitations, including the fact that they do not run in all installation scenarios and that they require the user to have Administrator credentials. When the current user does not have, or cannot obtain Administrator credentials, the co‑installer does not run. So it is essential to ensure that the device installs and runs correctly even if the co-installer does not run.

In addition, the operating system might not be aware of system changes that co‑installers perform. This can cause an increased chance that changes made by a co‑installer might not survive system changes such as a service pack installation or migration.

## Multifunction Device Guidelines

Multifunction devices have several unique Setup challenges. For a white paper that describes best practices for multifunction devices, see “Multifunction Printer Design Recommendations” in the WDK.

## Driver Distribution

We encourage manufacturers to distribute their drivers as widely as possible. Inbox drivers and those hosted on Windows Update have the advantage of being automatically available from Plug and Play. You should distribute your drivers on media that ships with the device so that users on earlier operating systems or without Internet access can make their device work.

### Driver Signing

The first step to distributing a driver over any channel is getting it appropriately signed. This assures users that the driver that they are installing is actually the driver for their device.

The Windows Logo Program includes requirements and testing guidelines to help you verify that your drivers are acceptable. Drivers that pass logo testing are returned through Winqual with a signature file. This driver can then be included on device installation media, a manufacturer Web site, or Windows Update.

Unsigned drivers cause many problems for end users because users with a limited user account cannot install them and other scenarios such as secure point-and-print fail. Distributing unsigned drivers is strongly discouraged. You should either fix drivers that do not pass the Windows Logo Program guidelines or submit them through the unclassified device program if the device has a limited audience and cannot be made to pass an existing device class.

Windows Vista® provides a mechanism for a driver vendor to sign a driver with a non-Microsoft signature authority. However, this mechanism requires end users (or their system administrator) to first install the root certificate so that the driver is considered signed. Therefore, we discourage wide use of self-signing and encourage using the Windows Logo Program.

### Windows Update

Windows Update driver distribution is a convenient extension of the logo-submission and driver-signing process. During the driver-submission process, the Driver Test Manager provides a check box that the submitter can select to automatically publish the newly certified driver to Windows Update. Publishing drivers to Windows Update improves user installation success on Plug and Play and is preferred over software-first installation because it does not rely on a specific user action to succeed.

### Inbox

Inbox drivers have the advantage of being available on all systems as soon as the operating system is installed. Partners with high market share and general-purpose print/MFP devices should try to develop drivers for the inbox driver program.

# Device Configuration

Device configuration is potentially a major source of end-user confusion. Ports that do not deliver the functions advertised can cause a customer to return the item to the store.

## Identical Ports that Expose the Same Functions

Devices that include multiple, identical ports should expose exactly the same functionality on each port. For example, an MFP with print, scan, and storage functionality that includes two USB-B connectors should support print, scan, and storage on both USB-B connectors. Do not construct devices so that only a subset of the functions is available on one connector and a different set is available on the other connector.

There are two exceptions to the port parity guideline. Devices that expose a PictBridge or maintenance port should label the port as such, but parity with other ports on the device is not expected. Devices with fax functionality should provide good labeling on modem ports to indicate which port is the input and which port is the output. These ports are not expected to have functional parity.

## Devices that Enumerate All Functions

MFPs should enumerate all functions of a device on the ports on which the technology is supported. For network ports, we also encourage enumeration of printing and scanning functions over Web Services on Devices (WSD). For example, an MFP with print, scan, and storage functionality and USB and Ethernet ports should enumerate each of these functions on each of USB and Ethernet.

If a technology is supported only on a subset of the device ports, this limitation should be well documented to reduce user confusion and prevent support calls.

## Autoconfiguration of Installable Options

If a device supports installable options, you should automatically configure the options so that they are immediately available without additional user or Administrator action. This greatly improves the usability of a device, especially in enterprise scenarios where updating the device configuration on every client is very tedious.

The WDK includes an example of how to implement autoconfiguration of a printer’s finisher, hard drives, and memory, but vendor-specific implementations are allowed. For more information, see “Auto-Configuration Samples” in the WDK.

## Continuing Operation after Device Errors Are Cleared

Printers should be able to continue normal operation after device error conditions such as a paper jam or an out-of-paper situation are cleared. We also recommend that the job in progress completes after the error condition is cleared.

Printers should also be able to recover from a surprise disconnect or reconnect from the host or the network, regardless of what the printer is doing at the time. The disconnect or reconnect should leave the system in a consistent state with the host able to submit the next print job. The device can abort the current print job in this scenario. However, power-cycling the device or Windows is a poor user experience that you should avoid under all circumstances.

# Driver Guidelines

Although user-mode printer drivers cannot cause blue screen failures, they can cause system instability, print server downtime, and poor performance. If you fully implement the following guidelines for development and testing, you can improve the robustness of your drivers.

## Driver Components that Operate Only in User Mode

Printer driver components should operate only in user mode. Kernel-mode components are more likely to contribute to system instability, and the print infrastructure might not let kernel-mode components load.

The spooler does not load a printer driver that contains kernel-mode components. Other blocks of code that are related to the printer driver and are often bundled in the same driver package (such as a port monitor) are not restricted in the same way. Generally, port monitors are complex and can easily expose stability and security holes in the system if they are not written correctly. We do not recommend that you develop your own port monitors and we do not ship third-party port monitors through the inbox driver program.

## Driver Components that Are Built in the Same Processor Architecture

The print subsystem in Windows is designed to host and execute printer driver components of the native hardware architecture. The print subsystem provides compatibility for Win32® applications that execute on 64-bit systems through the Windows on Windows 64 (WoW64) layer. Remember that 64-bit code never executes on a 32-bit client.

## Drivers Used Concurrently or from Multiple Processes

Printer driver components can be executed by the print subsystem in configurations in which the printer is used concurrently by multiple processes running in multiple user sessions such as configurations that use Microsoft Terminal Services or Fast User Switching.

To prevent failures, printer driver components must be robust to concurrent execution. Printer drivers should avoid specific operations including, but not limited to: communicating between components without specifying a user, job, or other unique identifier; assuming that different printer driver components execute in the same process; and reading or modifying a common data file such as a configuration file.

The Windows platform provides features to help address concurrency issues. These features include the following:

* For asynchronous notifications and user interface (UI) (Windows Vista and later versions), drivers, monitors, and print processors can use the ASync UI together with a shell plug-in to always display information in the correct user session.
* Data can be stored under HKEY\_LOCAL\_USER in the registry, but must be removed by the driver at uninstallation.
* Stored data that is more complex than a simple registry key must be stored in the local user directory and use the same file storage conventions as applications. This data must be removed by the driver at uninstallation. If an Administrator deletes a driver, all user data should be deleted by the driver during uninstallation.

## Driver Isolation

Print, scan, and fax device drivers are installed in many configurations on client and server machines. To ensure the correct operation of each driver, you should take special care during development and testing to ensure that a driver does not negatively interact with drivers for other devices or with different versions of the driver for the same device.

### Process Isolation

Printer driver components are executed in the same process as the print subsystem by default. The failure of a single printer driver component can cause the print subsystem to crash, which halts print operations for all users and all print components. To avoid this situation, printer driver components should test for, and report compatibility with, the Driver Isolation feature in Windows 7.

### Driver Installations that Do Not Prevent Other Drivers from Functioning Correctly

Imaging device drivers that install their own communication stacks must not prevent other drivers from using the Microsoft-supplied operating system stack.

Name driver files so that file names are unique to the manufacturer and consistent across driver packages. To avoid conflicts with other manufacturers, we recommend that you name driver files starting with the PostScript manufacturer code for the first two letters. Driver files of different types should not share the same name, either. For example, two files that are named ms000000.dll should provide the same functions so that the driver packages on which they rely can be installed concurrently.

## Unsupported Escapes and Events

Drivers should be robust and accept unsupported escapes and events. They should not crash, nor fail a print job if unsupported escapes or events are received. This enables a driver to exist on newer revisions of Windows if escapes or events are added.

## Port Monitors

This section provides guidance for using port monitors.

### Using Microsoft Port Monitors

Microsoft provides inbox port monitors for connectivity transport support between devices and Windows. Port monitor support is available for both network-connected and direct-attached devices.

#### WSDMon

WSDMon for printing is the preferred connectivity method for network-connected devices. WSDMon supports the Print Service Definition Version 1.0 for Web Services on Devices and acts as a median between devices, drivers, and the print spooler.

WSDMon supports increased bi-directional support in Windows Vista and later versions. Bi-directional support lets drivers and the print spooler query for device information that is obtained from the device by using WSDMon.

WSDMon is part of the Windows Rally™ technology program for network-connected devices. The adoption of WSDMon gives devices an advantage for discovery, Setup, and real-time event handling between devices and Windows.

#### TCPMon

TCPMon for printing is the legacy connectivity method for network-connected devices and will continue to be supported for Windows 7.

#### Third-Party Port Monitors

We recommend that you build drivers by using the inbox Microsoft port monitors and refrain from using third-party port monitor technology that must be packaged or downloaded on the system.

Third-party port monitors have been shown to cause instability with Windows client and server print spoolers.

## Core Drivers

This section provides guidance for using core drivers.

### Using Microsoft Core Drivers

Microsoft core drivers are designed to improve the agility in which drivers are created. They provide common rendering and configuration functions as well as a large amount of customizability.

#### Unidrv

You should always build Unidrv drivers by using the generic printer description (GPD) as the data file. It might be possible to store configuration data outside the GPD, but synchronization could prove tricky. Using as much of the provided technologies as possible also yields much greater compatibility on later versions of the operating system and can even provide extra functionality such as PrintTicket and PrintCapabilities support with little or no extra development effort.

For more information on Unidrv, refer to “Microsoft Universal Printer Driver” in the WDK.

#### PScript

Like Unidrv, PScript drivers should always be built by using the PPD as the data file.

For more information on PScript, refer to “Microsoft PostScript Printer Driver” in the WDK.

#### XPSDrv

XPSDrv drivers allow a tremendous amount of modularity and extensibility in the filter pipeline. However, you must carefully plan use of the filter pipeline to produce high-performing, robust drivers. Organize the filters so that minimal format conversion is necessary. Stream-to-document conversions are particularly expensive, so you should group stream-based filters at the end of the set of filters in use.

To avoid resource deadlock, global variables that are used in the filter pipeline must be safe for concurrent access.

#### Redistribution

We support redistribution of the latest Unidrv, PScript, and XPSDrv components to down-level versions of Windows. For more information on redistributing these drivers, refer to ”Redistributing Windows Vista Print Drivers” on the WHDC Web site.

### User Interfaces

Unidrv and PScript both provide user interfaces (UIs) automatically, but these can be extended. Drivers should provide a responsive UI at all times. When a worker thread calls high-latency APIs, use the thread to provide a responsive UI while it waits for the API call to complete. This action improves user perception of the performance of a dialog box, even if the API call is just as slow, because the UI remains responsive.

Rendering code must never create UI.

### Using Third-Party Core Drivers

Third-party core drivers can contribute to slimmer driver architectures when they are distributed inbox. All core drivers must be in the driver store for a model installation that depends on the core driver to succeed. Otherwise, installations can fail silently and offer the user little idea of what to do.

## Cluster Scenarios

Clustered print servers offer increased availability in high-volume environments. If your printer might be expected to exist in a clustered environment, you should carefully design your driver with cluster compatibility in mind. The biggest places where incompatibilities arise are with multiple print processor binaries and with custom port monitors.

You should combine multiple print-processor binaries into a single binary and reference them from the INF by using the PrintProcessor directive. For more details, refer to “Printer INF File Entries” in the WDK.

We discourage the use of custom port monitors, especially for devices that operate in a clustered environment. The **InitializePrintMonitor2** interface enables custom port monitors to be constructed to function correctly on a clustered print server.

MSDN® describes how to convert a port monitor that is not cluster compatible into one that can work with a cluster. Refer to “Converting Print Monitors for Use with Clustered Print Servers” in the WDK. The most common issues are related to using global variables and direct access to the registry.

## Backward and Forward Compatibility

Drivers that are written for Windows are frequently backward and forward compatible. Drivers that depend on a particular version of a Microsoft core driver such as Unidrv or PScript can redistribute them (as described in “Redistribution” earlier in this paper).

Drivers should avoid using operating system version checks that block installation on future versions of Windows. This practice can cause unnecessary driver compatibility issues and can require code changes as well as recertification.

Drivers that are developed on a later operating system should also try to be backward compatible. Operating system version checks are appropriate for drivers that have known compatibility issues that would result in an unacceptable user experience, but drivers should be provided to users as widely as possible.

# Security Guidelines

We recommend that all implementations follow published guidelines for reliability and security throughout the software development life cycle. For more information, see “Microsoft Security Development Lifecycle” in the MSDN library and “Reliability, Security and Maintenance” on the WHDC Web site.

## Working with Limited User Accounts

During driver development, it is important to consider Windows Vista’s increased use of Limited User Accounts. Using the Limited User Account is a considerable improvement in the security of a user system, but can cause failures that would not be encountered by an Administrator.

Drivers should be careful to check that any temporary directories that are being used can be written to at runtime. OEM versions of Windows might have already used directories such as “C:\Temp,” but not set write permission for users who have a limited user account.

Using co-installers during Setup can also cause issues for users who have a limited user account and would be required to obtain elevated privileges to install the contents of the co-installer. Users who have a limited user account can install signed, INF-based packages without elevated privileges. Therefore, signed, INF-based packages are the suggested method of driver distribution.

## Data Execution Prevention

Data execution prevention (DEP) is a technology that was introduced in Windows XP Service Pack 2 that prevents many security problems including buffer overruns. DEP can, however, terminate software with executable code pages that are not properly marked. You should always built and test drivers in an environment that has DEP enabled. For more details on how to build DEP-compliant components, see “Data Execution Prevention” in the MSDN library.

## Responsible Use of Personally Identifiable Information

Correct use of personally identifiable information (PII) is important to the security of our mutual customers. Printer driver structures or files such as DEVMODEs and PrintTickets are not secure and can be persisted in documents or spool files. Therefore, any PII that must be stored in those structures or files should be encrypted or hashed with a generally accepted encryption or hashing algorithm.

## Reliability on Secure Printers

Printers that implement secure printing features should include a mechanism to discard unprinted jobs after a stated time interval. Device Administrators should be able to edit this time to help accommodate projected device load. Drivers should also implement a failure case for when the device can no longer store new secure print jobs.

# Development and Testing Guidelines

## Development Guidelines

This section provides guidance on building printer drivers.

### Windows Driver Kit

The WDK includes several samples of how to build drivers, driver components, and even INFs. We strongly suggest that you build new drivers by using the latest sample drivers. These drivers are not guaranteed to be bug free or able to pass logo certification as is, but we strive to deliver very high-quality samples and to fix bugs regularly.

#### Setting Up a Build Environment

We do not support building printer drivers outside the WDK environment. Therefore, we strongly recommend that the only build environment that you use for driver development is the WDK environment.

For more information on the WDK build environment, refer to “Build Environment for Windows Drivers” on the WHDC Web site.

The WDK is available from MSDN subscriber downloads or from Microsoft Connect and is updated when new versions of Windows or Windows Server® are released, or for service packs.

## Testing Guidelines

This section provides guidance on testing printer drivers.

### Print Verifier

This section provides information about using the PrintVerifier tool.

#### What is PrintVerifier?

PrintVerifier is a runtime verification engine for monitoring interaction between third-party components and the print subsystem. In simpler terms, PrintVerifier helps resolve issues such as the following:

* How do I detect printer handle and general printer resource leaks in my application?
* How do I detect incorrect usage of Win32 printing APIs in my application?
* How do I detect incorrect usage of PrintTicket APIs in my application?
* How do I detect general printing-related bugs in my application?
* How do I detect incorrect reference counting in my printer driver?
* How do I detect incorrect implementation of printing-related COM methods in my printer driver?
* How do I detect incorrect PrintTicket or PrintCapabilities implementation in my printer driver?
* How do I detect bugs in my XPSDrv filter drivers?
* How do I detect all the previously stated problems at runtime?

PrintVerifier is implemented as an Application Verifier provider. It consists of two verification layers—the PrintAPI layer and the PrintDriver layer. As the names suggest, the PrintAPI layer monitors the boundary between applications and the print subsystem whereas the PrintDriver layer monitors the boundary between the core printer driver and third-party plug-ins.

Enabling PrintVerifier for your applications and the print subsystem gives you all the verification capabilities that were listed previously for free. For details on how to enable PrintVerifier or any other test that is part of Application Verifier, see “Using Application Verifier Within Your Software Development Lifecycle” on MSDN.

#### When to Use PrintVerifier

You should use PrintVerifier throughout your software development life cycle (SDL). The guidelines for when to use PrintVerifier are identical to the guidelines for Application Verifier. For Application Verifier details, see “Using Application Verifier within Your Software Development Lifecycle” on MSDN.

#### Additional Resources for PrintVerifier

For a wealth of information about PrintVerifier, see “PrintVerifier Team Blog” on MSDN. For a screencast of PrintVerifier in use, see “PrintVerifier demo” in the Microsoft communities.

### Windows Logo Kit

Microsoft ships several very important and useful tests in the Windows Logo Kit (WLK). The set of tests in the WLK is meant to validate the functionality of the driver in several specific areas such as robustness, rendering fidelity, PrintSchema conformance, XPS specification conformance, and WSD specification conformance. Although these are mandatory at the time of executing a logo submission, we recommend that printer driver developers run these tests during development to find potential issues early in the product life cycle.

### Using other Microsoft-Provided Test Tools

In addition to the set of tests that ships in the WLK, we also ship some useful tools in the WDK. The tools that currently ship in the WDK are PTConform, INFGate, IsXps, and LooksGood. These tools can be run directly from the command line and should be easy to integrate into a driver development environment.

# Resources

#### MSDN

Data Execution Prevention

<http://go.microsoft.com/fwlink/?LinkId=122801>

Microsoft Security Development Lifecycle

<http://go.microsoft.com/fwlink/?LinkId=125745>

PrintVerifier demo

<http://go.microsoft.com/fwlink/?LinkId=125751>

PrintVerifier Team Blog

<http://go.microsoft.com/fwlink/?LinkId=125750>

Using Application Verifier Within Your Software Development Lifecycle

<http://go.microsoft.com/fwlink/?LinkId=125747>

#### WDK

Auto-Configuration Samples

<http://go.microsoft.com/fwlink/?LinkId=122792>

Converting Print Monitors for Use with Clustered Print Servers

<http://go.microsoft.com/fwlink/?LinkId=125737>

INF Models Section

<http://go.microsoft.com/fwlink/?LinkId=125743>

Microsoft PostScript Printer Driver

<http://msdn.microsoft.com/en-us/library/aa506075.aspx>

Microsoft Universal Printer Driver

<http://go.microsoft.com/fwlink/?LinkId=125744>

Multifunction Printer Design Recommendations

<http://go.microsoft.com/fwlink/?LinkId=40504>

Printer INF File Entries

<http://go.microsoft.com/fwlink/?LinkId=125736>

Using Updated Core Print Drivers

<http://go.microsoft.com/fwlink/?LinkId=125742>

#### Windows Hardware Developer Central

Build Environment for Windows Drivers

<http://www.microsoft.com/whdc/devtools/tools/buildenv.mspx>

Redistributing Windows Vista Print Drivers

<http://go.microsoft.com/fwlink/?LinkId=122797>

Reliability, Security, and Maintenance

<http://go.microsoft.com/fwlink/?LinkId=125746>