Windows Driver Foundation: A Syllabus

A study guide for KMDF, UMDF, and Windows device driver concepts - April 2008

This article is a brief annotated list of resources for Windows® driver developers. These resources, together with many more that are not discussed here, are available from the WHDC Web site, the Microsoft primary information center for hardware and driver development.

**Midterm 1:** WinHEC 2007technical sessions and hands-on labs.

**Midterm 2:** Testing with the current release of the Windows Driver Kit (WDK).

**Final exam:** Your first KMDF or UMDF driver.

**Grading policy:** Self-assessment through Driver Verifier, PRE*f*ast, Static Driver Verifier, and WDK tests for drivers.

Basic Text

*Developing Drivers with the Windows Driver Foundation*, by Penny Orwick and Guy Smith

[www.microsoft.com/whdc/driver/wdf/wdfbook.mspx](http://www.microsoft.com/whdc/driver/wdf/wdfbook.mspx)

Provides comprehensive information about how to develop drivers that are based on the Windows Driver Foundation (WDF) model.

In this syllabus

[WDF](#_Toc196621045)

[KMDF](#_Toc196621046)

[UMDF](#_Toc196621047)

[General Windows Driver Concepts](#_Toc196621048)

[Tools for Driver Developers](#_Toc196621049)

[WDF and the WDK](#_Toc196621050)

# WDF

Windows Driver Foundation (WDF) is the Microsoft unified driver model. It supports the creation of object-oriented kernel-mode and user-mode drivers for Windows. By using WDF, driver writers can focus on their device hardware, instead of on the operating system. WDF simplifies driver development and maintenance in a number of ways, including:

Managing most interactions with the operating system.

Enabling a broader range of devices to be supported by user-mode drivers.

Supporting a robust, well-designed object model.

Providing intelligent default handling for common features such as Plug and Play and power management.

Reducing the occurrence of common race conditions.

WDF consists of two components, the kernel-mode driver framework (KMDF) and the user-mode driver framework (UMDF), which are used to implement kernel-mode or user-mode drivers, respectively. It also includes several related testing and debugging tools.

WDF home page on WHDC Web  
[www.microsoft.com/whdc/driver/wdf/default.mspx](http://www.microsoft.com/whdc/driver/wdf/default.mspx)

Introduction to the Windows Driver Foundation

[www.microsoft.com/whdc/driver/wdf/wdf-intro.mspx](http://www.microsoft.com/whdc/driver/wdf/wdf-intro.mspx)

Provides a high-level overview of WDF. It describes how the limitations of previous driver models led to the design and development of WDF and explains the overall goals and components of WDF.

Architecture of the Windows Driver Foundation

[www.microsoft.com/whdc/driver/wdf/wdf-arch.mspx](http://www.microsoft.com/whdc/driver/wdf/wdf-arch.mspx)

Provides detailed information about how WDF is imple­mented. It describes the overall architecture and explains how WDF can help reduce driver development time, contribute to greater system stability, and improve driver diagnosability and serviceability.

I/O Flow and Dispatching in WDF Drivers

[www.microsoft.com/whdc/driver/wdf/ioreq\_flow.mspx](http://www.microsoft.com/whdc/driver/wdf/ioreq_flow.mspx)

Describes the overall path of an I/O request and provides details about how KMDF and UMDF drivers process a request, from dispatch through completion. The framework manages the mechanics of dispatching, queuing, and canceling I/O requests on behalf of its drivers. It calls the driver’s event callbacks to notify the driver of significant events that the driver must handle.

Plug and Play and Power Management in WDF Drivers

[www.microsoft.com/whdc/driver/wdf/WDF\_pnpPower.mspx](http://www.microsoft.com/whdc/driver/wdf/WDF_pnpPower.mspx)

Describes how WDF drivers can support Plug and Play and power management by “opting in” to the extensive features and intelligent defaults that WDF provides.

Writing USB Drivers with WDF

[www.microsoft.com/whdc/driver/wdf/USB\_WDF.mspx](http://www.microsoft.com/whdc/driver/wdf/USB_WDF.mspx)

Describes how to use the USB I/O targets in both UMDF and KMDF to write a fully functional Windows driver for a USB device that uses the Windows USB device stack.

# KMDF

KMDF supports an object and runtime model that is used to implement kernel-mode drivers. KMDF drivers typically require little or no code for many standard operations, because they can use the default support that KMDF provides, especi­ally for Plug and Play and power manage­ment. KMDF drivers implement support for only those operations that are relevant to the operation of the associated device.

KMDF home page on WHDC Web  
[www.microsoft.com/whdc/driver/wdf/KMDF.mspx](http://www.microsoft.com/whdc/driver/wdf/KMDF.mspx)

Architecture of the Kernel-Mode Driver Framework

[www.microsoft.com/whdc/driver/wdf/KMDF-arch.mspx](http://www.microsoft.com/whdc/driver/wdf/KMDF-arch.mspx)

Provides a detailed overview of KMDF architecture and the types of drivers that can be developed with it. It covers the KMDF object model and shows how KMDF drivers handle I/O, Plug and Play/power management, and WMI requests.

DMA Support in KMDF Drivers

[www.microsoft.com/whdc/driver/wdf/dma.mspx](http://www.microsoft.com/whdc/driver/wdf/dma.mspx)

Describes how KMDF drivers support direct-memory access (DMA) devices. It explains the Windows DMA architecture and DMA abstraction.

An Introduction to How to Build, Install, Test, and Debug KMDF Drivers

[www.microsoft.com/whdc/driver/wdf/KMDF-build.mspx](http://www.microsoft.com/whdc/driver/wdf/KMDF-build.mspx)

Describes the basic mechanics of creating a KMDF driver. It does not discuss driver implementation, but covers the essentials of how to install KMDF and to build, install, test, and debug a simple KMDF driver.

Sample Drivers for the Kernel Mode Driver Framework

[www.microsoft.com/whdc/driver/wdf/KMDF-samp.mspx](http://www.microsoft.com/whdc/driver/wdf/KMDF-samp.mspx)

Steps through three sample KMDF drivers to explain their structure and show how to imple­ment common driver features. It is intended for driver writers who are familiar with KMDF and are preparing to use it to write their first kernel-mode driver.

How to Enable the Frameworks Verifier

[www.microsoft.com/whdc/driver/tips/KMDFVerifier.mspx](http://www.microsoft.com/whdc/driver/tips/KMDFVerifier.mspx)

Describes KMDF’s built-in verifier, which verifies correct I/O cancellation and queue usage, checks locking code, and ensures that KMDF and the driver follow the docu­mented contracts.

How to Use the KMDF Log

[www.microsoft.com/whdc/driver/tips/KMDF\_IfrLog.mspx](http://www.microsoft.com/whdc/driver/tips/KMDF_IfrLog.mspx)

Describes how to use the internal trace logger in KMDF. The trace logs track the progress of I/O request packets (IRPs) through the framework and the corresponding requests through a driver.

# UMDF

The UMDF object model is used to implement user-mode drivers for protocol-based or serial bus–based devices that previously required kernel-mode drivers. UMDF and KMDF drivers handle the same types of I/O, are installed by INF files, and have comparable performance.

However, UMDF drivers access only the user address space, posing a much lower risk to system security and stability than kernel-mode drivers. Other advantages include:

Simpler driver environment

Access to the Win32® API

Debugging with a user-mode debugger

Programming in C++

Rapid code generation

UMDF drivers have some limitations, however. Drivers that require the following must be kernel-mode drivers:

Handling interrupts

Direct hardware access, such as DMA

Strict timing loops

Use of nonpaged pool or other kernel-mode resources

UMDF home page on WHDC Web  
[www.microsoft.com/whdc/driver/wdf/UMDF.mspx](http://www.microsoft.com/whdc/driver/wdf/UMDF.mspx)

Introduction to the WDF User-Mode Driver Framework



[www.microsoft.com/whdc/driver/wdf/UMDF\_Intro.mspx](http://www.microsoft.com/whdc/driver/wdf/UMDF_Intro.mspx)

Provides a high-level overview of the UMDF driver architecture. It describes the advantages and features of UMDF drivers, and provides guidelines for determining whether to write a user-mode or a kernel-mode driver.

Architecture of the User‑Mode Driver Framework

[www.microsoft.com/whdc/driver/wdf/UMDF-arch.mspx](http://www.microsoft.com/whdc/driver/wdf/UMDF-arch.mspx)

Describes the details of the architecture and features of UMDF and outlines the requirements for UMDF drivers. It covers the UMDF object model, the flow of I/O requests through a UMDF driver, and the Plug and Play and power management features of the UMDF model.

Sample Drivers for the User-Mode Driver Framework

[www.microsoft.com/whdc/driver/wdf/umdf-samp.mspx](http://www.microsoft.com/whdc/driver/wdf/umdf-samp.mspx)

Provides an overview of the sample drivers that are included with UMDF. It steps through the Skeleton sample driver and discusses how to use the Skeleton sample as a template for driver development.

An Introduction to COM for UMDF Developers

[www.microsoft.com/whdc/driver/wdf/UMDF\_COM.mspx](http://www.microsoft.com/whdc/driver/wdf/UMDF_COM.mspx)

Introduces the subset of COM that is used in UMDF for driver developers who have little or no COM experience.

# General Windows Driver Concepts

These resources are relevant for all types of drivers. Many of these papers describe basic operating system features or best practices for driver development.

## Kernel-Mode Fundamentals

Common Driver Reliability Issues

[www.microsoft.com/whdc/driver/security/drvqa.mspx](http://www.microsoft.com/whdc/driver/security/drvqa.mspx)

Describes several common errors and suggests how driver writers can find, correct, and prevent such errors.

Scheduling, Thread Context, and IRQL

[www.microsoft.com/whdc/driver/kernel/IRQL.mspx](http://www.microsoft.com/whdc/driver/kernel/IRQL.mspx)

Describes how thread scheduling, thread context, and a processor’s current interrupt request level (IRQL) affect operation of kernel-mode drivers.

Locks, Deadlocks, and Synchronization

[www.microsoft.com/whdc/driver/kernel/locks.mspx](http://www.microsoft.com/whdc/driver/kernel/locks.mspx)

Describes how to use synchronization mechanisms to protect shared memory locations in kernel-mode drivers.

Handling IRPs: What Every Driver Writer Needs to Know

[www.microsoft.com/whdc/driver/kernel/IRPs.mspx](http://www.microsoft.com/whdc/driver/kernel/IRPs.mspx)

Describes how I/O works in Windows and how drivers should manage and respond to I/O requests.

Cancel Logic in Windows Drivers

[www.microsoft.com/whdc/driver/kernel/cancel\_logic.mspx](http://www.microsoft.com/whdc/driver/kernel/cancel_logic.mspx)

Provides guidelines for driver writers to determine when support for IRP cancellation is required and how to implement it correctly.

I/O Completion/Cancellation Guidelines

[www.microsoft.com/whdc/driver/kernel/Iocancel.mspx](http://www.microsoft.com/whdc/driver/kernel/Iocancel.mspx)

Describes how drivers can sup­port the applica­tion-initiated I/O cancellation feature in Windows Vista®.

Multiprocessor Considerations for Kernel-Mode Drivers

[www.microsoft.com/whdc/driver/kernel/MP\_issues.mspx](http://www.microsoft.com/whdc/driver/kernel/MP_issues.mspx)

Describes common issues that can occur on multiprocessor systems.

Memory Management: What Every Driver Writer Needs to Know



[www.microsoft.com/whdc/driver/kernel/mem-mgmt.mspx](http://www.microsoft.com/whdc/driver/kernel/mem-mgmt.mspx)

Describes how to allocate and use memory in kernel-mode drivers. It describes the types of memory that are available for driver use and the best ways to test for memory-related problems.

Increasing System Power Efficiency through Driver Support for Runtime Idle Detection

[www.microsoft.com/whdc/system/pnppwr/powermgmt/s0idle\_driver.mspx](http://www.microsoft.com/whdc/system/pnppwr/powermgmt/s0idle_driver.mspx)

Describes how to conserve power by powering down an idle device while the system is running.

## Security Topics

Windows Security Model: What Every Driver Writer Needs to Know

[www.microsoft.com/whdc/driver/security/drvsecure.mspx](http://www.microsoft.com/whdc/driver/security/drvsecure.mspx)

Describes how the Windows security model applies to drivers and explains what drivers must do to ensure the security of their devices.

Threat Modeling for Drivers

[www.microsoft.com/whdc/driver/security/threatmodel.mspx](http://www.microsoft.com/whdc/driver/security/threatmodel.mspx)

Provides guidelines for creating threat models for Windows drivers. A threat model is a way of categorizing and analyzing the threats to an asset. From a driver writer’s perspective, the assets are the hardware, software, and data on the computer or network.

Kernel-Mode Code Signing Walkthrough

[www.microsoft.com/whdc/winlogo/drvsign/kmcs\_walkthrough.mspx](http://www.microsoft.com/whdc/winlogo/drvsign/kmcs_walkthrough.mspx%20)

Provides a walkthrough of how to digitally sign kernel-mode software for x64 versions of Windows Vista. Kernel-mode software must be digitally signed to be loaded on x64-based versions of Windows Vista and later versions of Windows.

# Tools for Driver Developers

Driver development tools include utilities for debugging, testing, verifying, and installing drivers.

Developer Tools home page on WHDC Web  
[www.microsoft.com/whdc/devtools/default.mspx](http://www.microsoft.com/whdc/devtools/default.mspx)

## Driver Installation

Designing Driver Packages for Corporate Deployment

[www.microsoft.com/whdc/driver/install/deploy.mspx](http://www.microsoft.com/whdc/driver/install/deploy.mspx)

Describes how to create driver packages that are ready for deployment, do not prompt the user and do not require administrative privileges

Device Installation FAQ

[www.microsoft.com/whdc/driver/install/installFAQ.mspx](http://www.microsoft.com/whdc/driver/install/installFAQ.mspx)

Answers common questions about device installation on the Windows family of operating systems.

Plug and Play Device Driver Deployment in Windows Vista and Windows Server 2008

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

Describes Plug and Play driver deployment scenarios for Windows Vista and Windows Server 2008.

Uninstalling Drivers and Devices on Windows Vista

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

Describes how to uninstall drivers and devices in Windows Vista.

## Debugging and Tracing

Enabling Drivers with Event Tracing for Windows

[www.microsoft.com/whdc/devtools/tools/EventTrace.mspx](http://www.microsoft.com/whdc/devtools/tools/EventTrace.mspx)

Provides guidelines for driver developers to correctly instrument their drivers.

Debugging Tools for Windows Overview

[www.microsoft.com/whdc/DevTools/Debugging/default.mspx](http://www.microsoft.com/whdc/DevTools/Debugging/default.mspx)

Describes Debugging Tools for Windows, a set of extensible tools for debugging user-mode programs and kernel-mode drivers. The package includes WinDbg, a powerful debugger with a graphical interface and a console interface, and the console-based debuggers NTSD, CDB, and KD.

Improve Driver Debuggability

[www.microsoft.com/whdc/devtools/debugging/debugtips.mspx](http://www.microsoft.com/whdc/devtools/debugging/debugtips.mspx)

Provides tips to help ease the debugging process during both testing and retail use.

KMDF Debugging Extensions

[www.microsoft.com/whdc/driver/wdf/KMDF-dbgext.mspx](http://www.microsoft.com/whdc/driver/wdf/KMDF-dbgext.mspx)

Documents the debugging extensions that are included with KMDF. They can be used with the WinDbg or KD kernel debugger to obtain a variety of KMDF-related information.

## Static Driver Verifier

Static Driver Verifier (SDV) is a compile-time tool that explores code paths in a device driver by symbolically executing the source code.

SDV places a driver in a hostile environment and systematically tests all code paths by looking for violations of KMDF or WDM usage rules. SDV makes very few assumptions about the state of the operating system or the initial state of the driver, so it can exercise situations that are difficult to exercise by traditional testing.

SDV home page on WHDC Web  
[www.microsoft.com/whdc/devtools/tools/sdv.mspx](http://www.microsoft.com/whdc/devtools/tools/sdv.mspx)

Introducing Static Driver Verifier

[www.microsoft.com/whdc/devtools/tools/sdvintro.mspx](http://www.microsoft.com/whdc/devtools/tools/sdvintro.mspx)

Provides an overview of SDV with insight into the requirements and limitation for using it to verify kernel-mode drivers.

Static Driver Verifier for KMDF Drivers: WHDC Lab

[download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/SDV-KMDF\_Lab.docx](http://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/SDV-KMDF_Lab.docx)

Introduces the basic features of SDV and shows how to use it to examine a KMDF driver.

## PRE*f*ast for Drivers

PRE*f*ast for Drivers (Prefast.exe) is a static source code analysis tool that detects certain classes of errors not easily found by the typical compiler. PRE*f*ast for Drivers includes PRE*f*ast, a component that detects common basic coding errors in C and C++ programs, and a specialized driver module that detects errors in kernel-mode driver code.

PRE*f*ast analyzes C and C++ source code by stepping through all possible execution paths in each function and simulating execution to evaluate each path for problems.

PRE*f*ast home page on WHDC Web  
[www.microsoft.com/whdc/devtools/tools/prefast.mspx](http://www.microsoft.com/whdc/devtools/tools/prefast.mspx)

PREfast Step-by-Step

[www.microsoft.com/whdc/DevTools/tools/PREfast\_steps.mspx](http://www.microsoft.com/whdc/DevTools/tools/PREfast_steps.mspx)

Describes PREfast, which detects certain classes of errors not easily found by the typical compiler. It operates on each function in the source separately, pro­ducing a single combined list for all of the files checked in a single run, with duplicates elimi­nated.

PREfast for Drivers: WHDC Lab

[download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/PREfastForDrivers\_Lab.docx](http://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/PREfastForDrivers_Lab.docx)

Introduces PREfast through a series of exercises that use intentionally incorrect code to show how to run the tool and how to analyze some warnings that it produces.

# WDF and the WDK

WDF is available as part of the WDK. In addition, the WDK, WDF, and Windows Logo Kit (WLK) beta programs share space at a Microsoft Connect beta site.

You can download the current release of the WDK from MSDN Subscriber downloads. You can also join the WDK/WLK/WDF beta program and download the kit from the Microsoft Connect Web site.

WDK/WLK home page on WHDC Web  
[www.microsoft.com/whdc/devtools/WDK/WDKpkg.mspx](http://www.microsoft.com/whdc/devtools/WDK/WDKpkg.mspx)

To sign up for the WDK beta program

1. Go to [http://connect.microsoft.com](http://connect.microsoft.com/).

2. Sign in to the Web site.

3. Click Connection Directory at the top of the screen.

4. Scroll down to “Windows Driver Kit (WDK), Windows Logo Kit (WLK), and Windows Driver Framework (WDF)” and then click Apply Now.

You are automatically approved for entry.

5. Please fill out the survey before you download bits.