

# Writing WDF Drivers for Windows® (Accelerated Edition)

## Overview

The Windows Driver Foundation (WDF) is the latest standard for creating Windows drivers, and is the preferred way to implement most new drivers for Windows.

This fast paced, concentrated, seminar is designed for engineers who need to understand how to design, develop, or test Windows drivers using the WDF Kernel Mode Driver Framework (KMDF). The seminar focuses specifically on KMDF software drivers, filter drivers, drivers for Universal Serial Bus (USB) devices, and drivers for programmed I/O type Peripheral Connect Interface (PCI) devices. However, once you understand the basic concepts of WDF, creating KMDF drivers for other classes of devices and creating drivers using the User Mode Driver Framework (UMDF) should be relatively easy.

In this special accelerated seminar we cover more details about KMDF than in any other OSR seminar. We even add an complete additional module discussing extending KMDF to certain network (NDIS) and Audio/Video (Kernel Streaming) devices.

The seminar is designed for both developers who have no experience writing Windows drivers, as well as those who know how to write drivers using the Windows Driver Model (WDM) or who already have some experience with WDF.

Your learning continues even after the seminar is complete: For students who want to be able to immediately put what they learn into practice, a complete set of lab exercises including sample drivers, assignments, and solutions are provided to take home. USB and PCI lab assignments use the readily available and inexpensive OSR USB FX2 and Sealevel Systems 8018 Digital I/O kits, which are both available through the OSR Online store ([www.osronline.com](http://www.osronline.com))

## Seminar Formats

This seminar is a special accelerated 4 day lecture.

## Target Audience

Engineers who need to understand how to design, develop, or test Windows drivers using KMDF. Engineers who know already know how to write Windows drivers using WDM will be surprised at how easy KMDF makes their lives. After this seminar, engineers who've never

developed drivers for Windows will almost certainly wonder why everyone says writing Windows drivers is so hard (because, when OSR explains to you how to use KMDF, you'll find it isn't as hard as everyone says!).

**Note:** Regardless of location, this seminar is taught in English.

## **Prerequisites**

Students attending this seminar must have solid knowledge of general operating systems concepts (user mode versus kernel mode, virtual memory concepts, concurrency issues), device concepts (registers, interrupts), and the basics of Windows O/S Architecture. Previous experience developing device drivers (on any operating system) will definitely be an advantage.

## **Seminar Outline**

### **1. Introduction**

Welcome remarks, seminar goals and objectives, and a brief introduction to WDF and KMDF.

### **2. Windows Architecture Overview**

A brief review of Windows operating system architecture, focused specifically on the details needed by a KMDF driver writer.

### **3. The WDF Object Model**

WDF object characteristics and taxonomy. How objects are instantiated and used in KMDF. An overview of the most common WDF objects.

### **4. Driver Initialization**

How to initialize a KMDF driver and its associated device. Also, handling typical Plug and Play (PnP) and power management events such as device arrival, power-up, and power-down. How KMDF drivers are notified of and claim their hardware resources.

### **5. Driver Installation**

How to create installation control files for KMDF drivers. The Ten Most Frequently Used INF File Sections are discussed. The KMDF Co-Installer, and how to specify it in an INF file, is described.

### **6. Building and Debugging**

Using the Windows Driver Kit (WDK) to build KMDF drivers. How to setup the Windows Debugger (WinDbg), and a brief overview of the WDF Kernel Debugger Extensions (WDFKD), including retrieving the WDF Log from the "in flight recorder." Also a

discussion of various driver debugging tools relevant to WDF drivers (Driver Verifier, Static Driver Verifier, Prefast for Drivers).

## **7. The Windows Device Tree**

A description of how the Windows PnP subsystem discovers and enumerates drivers. All about Physical Device Objects (PDOs), Function Device Objects (FDOs), and filter devices. How filter drivers work their magic. How requests are processed, and passed from driver to driver within the Windows I/O Subsystem.

## **8. Interrupt Levels & Deferred Procedure Calls**

In this module, we discuss the all-important concept of Interrupt Request Levels (IRQLs), and the specific uses that Windows makes of various IRQLs. We also discuss Deferred Procedure Calls (DPCs) and how they're used in Windows for Interrupt Service Routine completion (DPCforISR).

## **9. Queues and Requests**

In this section, we discuss WDFQUEUES and WDFREQUESTS. Topics include how Queues are instantiated, Queue dispatch types, and how Queues can be used to sort Requests. We also discuss Framework Requests and how Requests are processed and completed.

## **10. I/O Targets**

Local, Remote, and Special I/O Targets are discussed. How to forward Requests both synchronously and asynchronously to other drivers in the system for processing. Completion routines are also covered.

## **11. Buffer Methods and IOCTL Definitions**

In this section, the different ways that requestor data buffers can be described are discussed. Direct I/O, Buffered I/O and "Neither I/O" are described, compared, and contrasted. Also discussed is how to define custom Device IO Control Codes (IOCTLs), and how the previously described buffering methods apply to IOCTLs.

## **12. Serialization**

In this module, we discuss issues relating to synchronizing access to shared data from within your driver. The much misunderstood topic of KMDF Synchronization Scope is fully described, as is extending sync scope to other callback routines via the Automatic Serialization parameter. WDFSPINLOCKS and WDFWAITLOCKS are discussed, along with the underlying implementations of each and how they're used.

## **13. USB Concepts**

The basics of USB are discussed including device, configuration, and interface descriptors. Endpoints and pipes are described.

## **14. Implementing WDF USB Drivers**

In this section, we describe how USB drivers are implemented in KMDF. This includes how a configuration and interface is chosen, and how endpoints are retrieved. How to send vendor commands to a device via Endpoint 0. Using Bulk and Interrupt endpoints. The WDF

Continuous Reader is briefly discussed, as is supporting Selective Suspend (USB device power management).

### **15. Implementing WDF Drivers for Programmed I/O Devices**

In this section we extend our discussion of receiving and processing requests to how drivers handle requests for programmed I/O type PCI and PCI Express devices. We describe how to implement your Interrupt Service Routine (ISR) and Deferred Procedure Call for ISR Completion (DPCforISR) in a KMDF driver.

### **16. Cleanup, Close and Cancel**

Strategies for handling queued and in-progress requests are discussed, as what processing typically takes place as part of cleanup and close processing.

### **17. Helpful Classes**

A brief description of a few Framework classes such as WDFCOLLECTION, WDFWORKITEM, and WDFTIMER that might be useful additions to your "bag of tricks."

### **18. Extended Driver Types**

Advantages of using the KMDF model in parallel with other Windows driver architectures. Implementation techniques for using KMDF drivers for certain NDIS or KS (kernel streaming) implementations.