

# Writing WDF Drivers for Windows<sup>®</sup>

## (Lecture with Lab Edition)

### Overview

This seminar is designed for engineers who need to understand how to design, develop, and test Windows drivers using the Windows Driver Foundation's (WDF) Kernel Mode Driver Framework (KMDF). While the seminar provides the groundwork and concepts for writing any supported type of WDF driver, its focus is providing practical experience in developing KMDF software drivers, filter drivers, and drivers for USB devices. Once an attendee understands these concepts, creating KMDF drivers for other classes of devices and creating drivers using the User Mode Driver Framework (UMDF) should be relatively easy.

Hardware Included A valuable part of your learning experience in this course will be the lab exercises targeted for use with the OSR USB FX2 Learning Kit (check out the OSR Online Store at [www.osronline.com](http://www.osronline.com) for more information.) Of course, we want your learning experience to extend beyond the classroom, so each student will walk away with their very own OSR USB FX2 Learning Kit.

### Seminar Formats

This seminar is available as a 3 day lecture, or 5 day lecture with interspersed hands on labs, which can be customized with additional or different topics for presentation at your location. Please contact an OSR Seminar Coordinator for details on arranging an on-site seminar.

### Target Audience

Engineers who need to understand how to design, develop, and test Windows drivers using the Windows Driver Foundation's (WDF) Kernel Mode Driver Framework (KMDF).

Note: *Writing WDF Drivers for Windows* is most appropriate for developers focused on writing new drivers for which WDF is applicable. Developers who seek a thorough understanding of the workings of the Windows I/O subsystem in preparation for writing non-WDF drivers are strongly urged to attend OSR's *Writing WDM Kernel Drivers for Windows* seminar instead of this seminar.

### 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.

Due to the hands-on orientation of this seminar, attendees will be assumed to be able to use Windows at a user level, including using Microsoft's Developer's Studio (VC++). Working knowledge of the C programming language, and how to read and write to a file using Win32 (CreateFile, ReadFile, WriteFile) are also assumed.

## **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 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.

### **4. 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.

### **5. Building and Debugging**

Using the WDK to build KMDF drivers. How to setup 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).

### **6. 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.

### **7. Driver Initialization**

How to initialize a KMDF driver and its associated device. Also, handling typical 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.

**Lab:** Building and Debugging, Driver Initialization (DriverEntry, EvtDeviceD0Entry, etc).

## 8. Interrupt Levels & DPCs

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.

**Lab:** Request Processing and Completion, Filtering

## 11. Buffer Methods and Device Controls

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. USB Concepts

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

## 13. 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).

**Lab:** USB, Using the OSR USB-FX2 device

## 14. 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.

## **15. 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.

## **16. 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."

**Lab:** USB continued, Open/Close processing