

## Windows Device Drivers ( WDM + WDF combined) 6 days

### Developing WDM Device Drivers For Windows

This is an intensive program intended at explaining the core concepts involved in Windows Driver development using the Windows Driver Model (WDM) framework. It is intended for device driver and device firmware writers for writing function/client drivers for devices such as audio, parallel/serial port and other devices.

### Developing WDF Device Drivers For Windows

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 engineers 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.

#### Pre-requisites:

- A. Sound knowledge of C programming (Must)
- B. Knowledge of Windows OS Internals (Must)
- C. Basic H/W device concepts such as memory mapping, port access, device registers etc. (Must)
- D. Communication protocols (Desirable)

The key learning from the program are:

- A. Structure of WDM drivers
- B. I/O Request data structures, handling and management
- C. Introduction to Plug'n'Play

#### Course Details:

### Developing WDM Drivers

#### Day 1 Basic Structure of a WDM Driver

- How Drivers and Applications Work
- Introduction to basic concepts:
  - Device and Driver Layering
  - Plug and Play Devices
  - Legacy Devices
  - IRP Routing
- The Two Basic Data Structures
  - *Driver* objects
  - *Device* objects
- Overview of *DriverEntry*, *DriverUnload* and *AddDevice* Routines
- Creating a Device object
- Naming Devices

#### Day 2 The I/O Request Packet Data Structures

- Structure of an IRP
- The I/O Stack The "Standard Model" for IRP Processing
- Creating an IRP
- Forwarding to a Dispatch Routine
- Duties of a Dispatch Routine
- The *StartIo* Routine
- The Interrupt Service Routine
- Deferred Procedure Call Routine
- I/O Completion Routines
- Advanced Request Processing
  - Queuing I/O Requests
  - The DEVQUEUE object
  - Cancelling I/O Requests
- Completing the Dispatch Routine

### **Day 3 Plug and Play for Function Drivers**

- Reading and Writing Data
- Configuring Your Device
- Addressing a Data Buffer
- Specifying a Buffering Method
- Ports and Registers
- Port Resources
- Memory Resources
- Servicing an Interrupt
- Configuring an Interrupt
- Handling Interrupts
- Deferred Procedure Calls

## **Developing WDF Drivers**

### **Day 1 – Introduction to the WDF**

- Review of WDM driver architecture
- WDM Concepts for WDF
  - Driver Types
  - Device and Driver Stacks
  - I/O Request Packets (IRPs)
- Quick Introduction to WDF
  - Kernel Mode Driver Framework (KMDF)
  - User Mode Driver Framework (UMDF)
  - Differences between KMDF and UMDF
  - When to use KMDF and UMDF?
  - Examples of devices suitable for KMDF and UMDF
- Requirements of A Simple WDF Driver
- Code walkthrough of WDM and WDF drivers for comparison

### **Day 2 – The Kernel Mode Driver Framework**

- Interactions between the KMDF and drivers
- Introducing the Basic KMDF Objects
- Framework Object Lifecycle

- The Object Context Space
- Handling I/O Requests
- Processing an I/O Request

### **Day 3 – The User Mode Driver Framework**

- Introducing the UMDF and benefits of UMDF drivers
- Supported devices and device classes
- Key Elements – Driver host process, Reflector, Driver Manager
- UMDF Objects and Object Hierarchy
- Managing UMDF Object Lifecycle
- Initializing UMDF drivers
- Callback Objects
- Processing an I/O request