

## Developing File System Minifilters for Windows

File systems on Windows are deeply integrated with the operating system. This integration is: filled with legacy edge cases, constantly evolving to support new Windows features, and (sadly) not particularly well documented. This makes file system development extraordinarily difficult for new developers and creates a large ongoing maintenance burden.

One might think that writing a file system filter would be a much easier task. In fact, over ten years ago the Filter Manager Minifilter model was introduced to make writing file system filters easier. There are even lots of starter samples available on [GitHub](#). So, developing a Minifilter should be “not so hard”, right? Well, that’s exactly what many developers think. At first.

But, here’s the truth: writing useful file system filters that work in the real world is hard. Really hard. In our experience, filters can be even harder to write than a typical file system. While the Minifilter model makes filters look easy, it does little to shield you from the complexity of the overall Windows file system environment. This leads to many file system filter developers finding themselves over their heads when they realize that their “easy” task is not so easy after all.

And that’s where we come in. We’ve been writing file systems and file system filters for Windows for over 20 years. In this seminar, we’ll teach you what you need to know to successfully write a Standard Windows file system Minifilter. And we’ll do it in a way that’s engaging, fast paced, and filled with both architectural insights and practical hints and tips.

*At the end of the seminar, students will understand the steps for implementing or maintaining a Standard Windows file system Minifilter driver. The requirements for and issues related to Isolation Minifilters are discussed, but this seminar is not designed to address Isolation. For more information about Standard versus Isolation Minifilters, please see [An Introduction to Standard and Isolation Minifilters](#).*

### Details

**Length:** 5 days

**Format:** Lecture and Lab

### Target Audience

This course is targeted specifically at developers needing to build or support a Standard Filter Manager Minifilter driver. Products that require the use of a Standard Filter Manager Minifilter include but are not limited to:

- Activity Monitors (e.g., Process Monitor)
- Endpoint Protection (e.g., antivirus)
- Hierarchical Storage Managers (HSM)
- File Backup/Replication
- And *many* more...

## Prerequisites

Students attending this seminar will be assumed to have a good working knowledge of general O/S concepts (user mode versus kernel mode, virtual memory concepts, and concurrency issues). Previous experience developing kernel mode software (on any operating system) will definitely be an advantage, but is not required.

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

## Hardware Requirements

Each student will need to bring a preconfigured and tested laptop with them to the seminar. This laptop will be used for doing lab assignments.

The hardware and software requirements for this seminar are described on our OSR Seminar Laptop Setup Requirements page.

If you have any questions regarding the required configuration, or it is not possible for you to provide a suitable laptop, don't hesitate to get in contact with our seminar team: seminars at osr.com. We'll be happy to help.

## Seminar Outline (Last revised 13 August 2018 — Still subject to change)

- **Introduction**
- **Windows OS Architecture Overview**  
A brief review of the general architecture of the Windows operating system, focusing on what you need to know as a file system minifilter developer.
- **File System Filtering Concepts**  
A brief discussion of basic Windows file system filtering concepts, a view into the world before Filter Manager was invented, and the types of problems (such as layering, dynamic loading/unloading, stack usage) that Filter Manager was designed to solve.
- **Introduction to Filter Manager and Minifilter Concepts**  
An introduction to core minifilter concepts, including altitudes, objects, callbacks, and context.
- **Windows Storage**  
If you're going to filter the file system, it's important to understand the overall concepts that are part of the storage "branch" of the Windows device tree. In this module, we discuss how the file system portion of the storage branch is instantiated, and how I/O requests are processed by file systems and filters. We discuss IRPs, I/O major and minor function codes, and Fast I/O. We then relate these concepts to minifilters, as we discuss Callback Data, PreOperation and PostOperation callbacks.
- **MiniFilter Initialization**  
In this module, we discuss the initialization that every minifilter undertakes as part of its DriverEntry processing. We also delve into the InstanceSetup callback and we delve further into the details of common PreOperation and PostOperation callbacks. We walk-through brief code segments for common routines.
- **Minifilter Installation**  
What you need to know to enable you to create installation control files for Filter Manager minifilters.
- **Building and Debugging**  
Students are assumed to already be familiar with the basics of how to use Visual Studio. We'll review the

details, as well as discuss how to setup and use WinDbg, the Windows kernel debugger. We also discuss the Filter Manager Debugger Extensions (FLT KD).

- **Caching and Virtual Memory Concepts**

The Windows Memory Manager provides the ability to memory map files and cache the retrieved data in memory. The Windows Cache Manager provides a single, common cache for all file data and file system metadata accessed via the standard read and write interfaces. Windows file systems must explicitly integrate with the Memory and Cache Managers. A discussion of the integration and the resulting access patterns. Memory mapped I/O. User I/O versus Paging I/O. Cc, Mm, and file system locking are discussed as is the concept of top level requests.

- **Interrupt Request Levels**

Windows synchronizes kernel mode activity by using a set of Interrupt Request Levels (IRQLs). This section covers how IRQLs are used to achieve synchronization within the OS. Also the processing that occurs at these IRQLs – including Deferred Procedure Calls (DPCs) and dispatching – are discussed.

- **Create/Open Processing**

How minifilters process CreateFile requests, including the differences between pre-create and post-create processing. We also discuss Extra Create Parameters, as well as the issues and methods of blocking create requests.

- **Read Processing**

How minifilters process read requests, including those from the standard ReadFile API as well as through memory mappings. The differences in handling user, paging, and paging file reads.

- **Write Processing**

How minifilters process write requests, including those from the standard WriteFile API as well as through memory mappings. The differences in handling user, paging, and paging file writes. Introduction to the Lazy Writer, the Modified Page Writer and the Mapped Page Writer.

- **Cleanup and Close Processing**

The difference between the cleanup and close operations, including common processing performed in each. Why your minifilter may never (literally) see a close for any given file.

- **Delete Processing**

The different ways in which a file may be deleted in Windows. A discussion of “delete as an intention” and the inherent difficulty in knowing when a file is actually deleted.

- **Rename Processing**

The different ways in which a file may be renamed in Windows. A discussion of the meaning and usage of the SL\_OPEN\_TARGET\_DIRECTORY create flag.

- **Opportunistic Locking (Oplocks)**

Oplocks began as a way to control caching on the network. However, their usage has grown significantly over the years and they are very commonly used for local file access as well. Oplock use cases are discussed, as well some of the common troubles caused by oplocks.

- **Dealing with File Names**

It is logical to want to assign file names with file system operations. However, naming is a shockingly complex topic. A discussion of how names are retrieved and why it is sometimes unsafe to do so. Normalized names versus opened names are discussed. Filter Manager Name Provider callbacks are discussed.