ISSUE VOLUME
EX/21
WINHEC
SHENZHEN
2015

The NT Insider

A publication of OSR Open Systems Resources, Inc.

WinHEC Extra Edition!

winhec 深圳 2015

What's New For Driver Developers From WinHEC Shenzhen 2015

Don't Miss The March-April Edition of The NT Insider **HERE**!

Published by
OSR Open Systems Resources, Inc.
105 Route 101A, Suite 19
Amherst, New Hampshire USA 03031
(v) +1.603.595.6500
(f) +1.603.595.6503

http://www.osr.com

Consulting Partners
W. Anthony Mason
Peter G. Viscarola

Executive Editor
Daniel D. Root

Contributing Editors
Scott J. Noone
OSR Associate Staff

Send Stuff To Us:
NtInsider@osr.com

Single Issue Price: $15.00

Still Crazy After All These Years
Some Progress, Some Change

Included in this Extra Edition of *The NT Insider*® are **updated versions of our blog posts** describing some of the new features announced at WinHEC Shenzhen 2015. If you read the blog posts as they were written, this is a chance to skim for updates to the original content. If you didn't get a chance to read those blog posts, now's your opportunity to catch-up on all the WinHEC news.

You can stay up to date by following us on Twitter (@OSRDRIVERS) or by joining the OSRHints mailing list (just send a blank email to join-osrhints@lists.osr.com).

You might think that because Windows has been around for a while, when a new version of Windows is released there wouldn't be much new in the world of driver development. For the last several years, nothing has been further from the truth.

Just THINK about the changes that we've seen over just the past few Windows releases:

- Integration of the WDK with Visual Studio
- New sets of driver samples
- Wizard-based driver development
- Support for new buses and peripheral types
- Introduction of UMDF V2, making it simple (if not entirely trivial) to move drivers from kernel-mode to user-mode.

And the **progress** continues. Win 10 brings mobile, IoT, and desktop/server development together into a single kit. The WDF Framework source code, and the WDK sample drivers, have joined .Net Core and the "Roslyn" compiler on GitHub. If you hated writing ASL or INF files, there are new tools to help with these things. You can add your own trace messages to the WDF Log. And Windows 10 has announced support for several new, little bitty SBCs. These things are clearly progress. Betterment. All positive innovations. I think we'd all agree.

Additional **changes** also continue to be introduced. "Push a button and deploy and test your driver" has once again been promised. And, as I'm sure many of you have already heard, Kernel-Mode drivers on Windows 10 will require Microsoft signatures to load. And getting that signature is going to require an Extended Validation (EV) certificate. Whether these changes constitute **progress** or not remains to be seen. We're *particularly* concerned about the wisdom and impact of requiring EV certificates… but time will tell.

If you're involved in the design, development, or testing of Windows drivers — for phone, tablet, desktop or IoT — what's most important is that you keep up to date with what's happening and get involved with the community. As we predicted almost a year ago, Microsoft is genuinely working hard to listen to customers and developers. We can make a difference. Get engaged. Join and participate in our NTDEV peer help list. Let your voice be heard.

You can see that, just because Windows has been around a while, this does NOT mean that big changes can't happen. They can… because we've seen them happen with each release, as a direct result of community feedback. And you can make sure that we don't just seen **change**, but that we actually see **progress** in areas that matter.

Stay tuned. I have no doubt there are still some surprises left to be announced between now and when Windows 10 is released.

Peter
@OSRDrivers

Return of the MVP Program
New MVP Program Announced for Hardware/Driver Devs

Back and the end of January we told you that Microsoft was reviving their interest in community. But I'm not sure anybody saw *this* particular change coming.

During WinHEC Shenzhen 2015, Microsoft introduced a new Most Valuable Professional category, called the *Microsoft Windows Hardware Engineering MVP Program*. This category recognizes engineers who develop software for electronic, electro-mechanical, or integrated circuitry for consumer hardware systems. Being an MVP in this category requires solid knowledge of the WDK as well as WDF.

While there *is* a File System Storage MVP category (and several well-known kernel mode devs are members including myself and OSR's Scott Noone), there hasn't been a Microsoft MVP category that recognizes device engineers, component developers, and driver developers who work on a wide array of devices since the Driver and Kernel Development Program was ended in 2011.

There are three things that make the Hardware Engineering MVP Program particularly interesting:

- The program has a distinctly international emphasis. With hardware innovation occurring world-wide, it makes sense to have world-wide representation in the program.
- The program is aimed at device, driver, and component developers who work on devices ranging from peripherals and component parts, to smart phones, tablets, laptops, desktops, servers and embedded systems. It also encompasses makers from both the commercial and educational sectors. So, it's unusually broad in its scope.
- The program promises to be more than just an honorary title or a bunch of Microsoft marketing hoopla. Microsoft has specifically asked for the MVP members to provide actionable feedback and "contribute to strategic business decisions regarding Windows hardware development."
- The program was announced during WinHEC in Shenzhen today. "It's about community, community, community" one of the MSFT program managers told me in a phone call. The MVPs who were in attendance at WinHEC were treated Wednesday evening to a special invitation-only event recognizing them and proving them a chance to interact directly with key MS executives. Don Box (well known writer and Microsoft Distinguished Engineer) spoke to the MVPs.

The inaugural members of the Windows Hardware Engineering MVP Program were drawn from the global community. Given the location of WinHEC this year and Microsoft's recognition of the importance of China and Taiwan as hubs of emerging technology, there is a particular emphasis on members from those geographical areas. New MVPs can be named at any time during the year as warranted.

The term for the new MVPs will officially begin on 1 April 2015 (the start of the new quarter). The Microsoft owner of the Windows Hardware Engineering MVP Program is *Asobo Mongwa*.

Congratulations to the inaugural members of this program!

Microsoft Windows Hardware Engineering MVP Program

Inaugural Members

(Listed in alphabetical order within geographical area)

North America	China	Taiwan
Don Burn	Bingjing Ge	Sam Hsieh
Brian Catlin	Deqiang Han	Andy Huang
Scott Noone	Hongfeng Liu	Chao-Jen Huang
Tim Roberts	Jiong Shi	Chun-Liang Liu
Peter Viscarola	Jacky Wang	Winfred Lu
	ShouBin Yang	
	Raymond Y Zhang	
	Jiewen Zheng	

Follow us!

Windows Driver Foundation Source Code Released
Windows Source Code on GitHub!

[WDF Source Available: https://github.com/Microsoft/Windows-Driver-Frameworks]

OK, well, maybe that title was a bit misleading. But, it's not *entirely* untrue: portions of the Kernel and User Mode Driver Frameworks are available on GitHub right now, even as you read this, for the benefit of the driver development community. The magnitude of this announcement cannot be understated as it represents a major paradigm shift in Microsoft's engagement with the development community. Not only is the source code on *freakin' GitHub*, but we're also getting private PDBs for the Framework binaries. This will enable us to perform source level debugging of the Frameworks in the comfort of our own offices, which is more than we imagined in our wildest dreams.

The version of WDF that was released is Build 10041 of WDF V1.15 — Microsoft has promised updates will be provided.

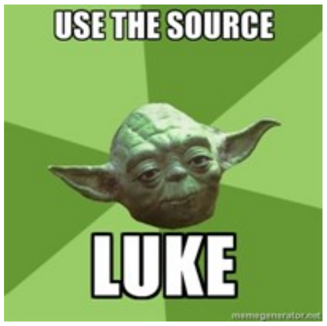
To understand why this news is a bit of a shock, we really need to go back in time to when the driver Frameworks were first introduced…

During the development of these Frameworks, Microsoft seriously engaged with the driver development community. Developers were able to provide feedback through forums, email aliases, and even an NDA'd, invite only conference on campus at Microsoft. The major sticking points brought up by the community during this time were the following:

1. The interface to KMDF had to be C based (hey, this was 2003)
2. The Frameworks had to support Windows 2000 (did I mention this was 2003?)
3. The Frameworks had to be open source

KMDF does indeed have a C based interface, so we won there. KMDF 1.0 shipped without Windows 2000 support and the community went apoplectic, leading to Microsoft reversing that decision with KMDF 1.1.

The source code issue was a sticking point for quite a while. In fact, in 2007 we published an article in The NT Insider titled *18 Months Later: Release The KMDF Source Code!* Also in 2007, there was a lengthy exchange on NTDEV with all of the hallmarks of a religious battle (including one WDF team member almost being able to "see the spittle"). The request for source code was clearly heard, but nothing came of it. We all moved on and adopted the Frameworks anyway, which is a testament both to how bad WDM was and how good the Frameworks are.



What you might *not* know is that WDF team wanted the community to have source code from the beginning. They agreed that having source would be a good thing and pushed for this internally at Microsoft. The decision to not release the source was, unfortunately, ultimately a business decision and not a technical one.

A decade later, here we are *with the source*. The fact that the change came *now* is both a testament to the WDF team's tenacity in its commitment to the community and the overall change that's happening at Microsoft in terms of community engagement.

As a community, we driver developers need to step up and show that this is the way that it should be. Show that we know how to, and will, make good use of this new tool. Microsoft will be accepting bugs submitted against the Framework through GitHub, so be sure to file a bug instead of complaining! Eventually the plan is to even allow for code contributions from the community, though the details of how/when that will happen are TBD.

Try Windows 10. Try the WDK 10. Submit bugs and feedback. Hopefully it won't take us 10 years to get everything we want, but even if it does we can now be sure that someone's listening!

Follow us!

One Driver Works Everywhere
Universal Drivers

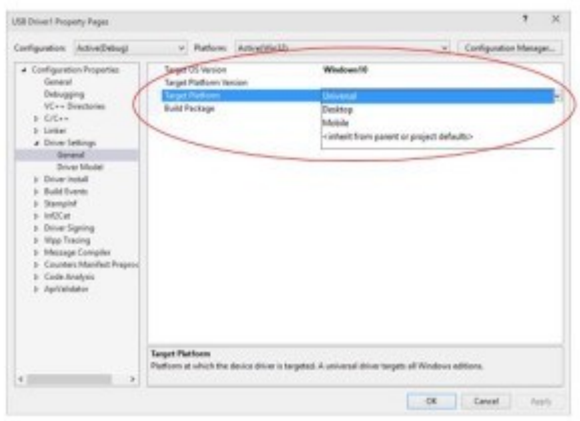
Windows 8 ushered-in a new era of Windows mobile devices, including phones and tablets. Windows 10 promises to converge these systems further, and to extend Windows to even more resource and cost constrained devices and the Internet of Things (IoT). Support for x86 and ARM will span all these categories. It seems we really are on the brink of having Windows Everywhere.

No, no, no…Universal Driver.

At other conferences such as Tech Ed, we've already heard talk of Universal Apps that you'll be able to build once per architecture, but run across all editions of Windows. But, until now we hadn't heard anything about how this applies to drivers.

At WinHEC in Shenzhen today, Microsoft announced support for **Universal Drivers**. Like Universal Apps, these drivers will run on all Windows editions (Desktop, Mobile, IoT). You build one binary per architecture. You can test on any edition of Windows, and still be reasonably confident that your properly written Universal Driver will work on all editions of Windows.

How do you know that your driver will work on multiple editions of Windows without testing? Well, most kernel-mode drivers are fully backwards compatible. For user-mode drivers, the WDK includes a new tool (APIValidator.exe) that will run as part of the post build process for universal drivers. APIValidator flags APIs that are used in the driver project that aren't part of the Universal platform. Still, if it was me, I'd want to do at least some small amount of testing on each platform to be completely sure.



Configuration: Active(Debug)
Platform: Active(Win32)

Target OS Version: Windows10
Target Platform Version
Target Platform: Universal
Build Package
Desktop
Mobile
<inherit from parent or project defaults>

Target Platform
Platform at which the device driver is targeted. A universal driver targets all Windows editions.

Target Platform is selectable in Windows 10

To support the development of Universal Drivers, the WDK has been significantly enhanced. Starting in Windows 10, there is **one WDK for all Windows driver development**. That single WDK will support the development of drivers targeting x86/x64 and ARM processors and all editions of Windows. To create a Universal Driver, you can just select "Universal" for your driver's Target Platform. The WDK will do the rest.

The Windows 10 WDK will work with Visual Studio 2015, and both the WDK and Visual Studio will continue to be free for driver development. And, according to the WDK PMs, it will also be possible to install and support multiple versions of the WDK side-by-side (though we haven't seen this feature in operation, and we're not exactly sure how it will work).

Follow us!

WHCK Becomes WLK
Updates to Driver Testing — Not Just for Certification

Long ago, Microsoft created the Windows Hardware Quality s (WHQL, pronounced “wickle”) program. The goal of this program was to certify that a particular device and its associated driver met a set of WHQL defined requirements. The requirements could include anything from how the driver was developed (e.g. which driver model it used) to whether or not the device and driver could survive the system being transitioned into and out of Standby multiple times. If your device and driver passed these requirements, your driver would be signed by Microsoft and you would be allowed to affix the “Designed for Windows” logo to your packaging. For this reason, we often referred to going through WHQL as “achieving logo” or “logo-ing”.

WHQL had two major downsides:

1. It wasn’t sufficiently transparent. When you submitted your driver for logo, you either passed or you didn’t. Also, it wasn’t clear at which point in your development process you were eligible for logo. As submissions weren’t free, it wasn’t in your interest to simply *attempt* to get a logo and see where you stood in the process.
2. The WHQL process happened at Microsoft’s campus. You had to physically ship them your hardware so they could validate it against their requirements. They were also under no obligation to send it back, which was, shall we say, inconvenient.

These downsides weren’t a mystery and Microsoft did work to make this process easier. Thus, WHQL was effectively scrapped for the Windows Logo Program. The big change with this new program was the ability to logo your device and driver using the Windows Hardware Compatibility Test (HCT). No more shipping stuff off to Microsoft, you could achieve logo in the comfort of your own office!

The HCTs were a nightmare to set up and use, so they were replaced with the Windows Logo Kit (WLK) for Windows Vista. The WLK was a nightmare to set up and use, so it was replaced with the Windows Hardware Certification Kit (WHCK) for Windows 7. Given that it’s now a *Certification Kit*, we also now certify devices and drivers instead of logo-ing them.

Needless to say, all of this turmoil turned off the driver development community from using any of these kits unless they absolutely had to. In this case, “absolutely had to” meaning that their customers actually cared about the Designed for Windows logo. This is unfortunate because the certification process is designed to find bugs and Windows incompatibilities in devices and drivers, why *wouldn’t* we want to go through that process in the hopes of increasing quality in our products?

To that end, with Windows 10 Microsoft is taking steps to make it easier to run the certification tests early and often. The first step is to kill off the WHCK and introduce the Hardware Lab Kit (HLK). This is said to be an “evolution” of the WHCK, so we can hope for continued ease of installation and use. Oh, and the testing isn’t called “certification testing” anymore… it’s called “compatibility testing.”

We’ll update on that when we’ve had a chance to play with the HLK more. One neat addition though is the ability to control the HLK process through C# and PowerShell, which should make automation possible. Unfortunately, the HLK will only be used to test drivers for Windows 10 and later, which means that you’ll still need a WHCK environment for older platforms.

The second step is more interesting still: It is now not only possible, but practical, to run a subset of the compatibility tests that are bundled with the HLK directly from your development system. At least that’s what the presentation at WinHEC in Shenzhen promised.

Microsoft recognized the desirability of getting driver devs to run a few tests right from their desks since Windows 8, when the WDK and Visual Studio were first combined. It’s always been a goal to “just press F5” to rebuild your driver, lob it onto a target system (a real system in your office or a VM, depending on the type of driver you’re writing), have it installed automatically, and start a series of basic tests. It MIGHT be possible to do this today with the Windows 8.1 Driver Kit, if you set things up right and everything works. Peter swears he’s seen it work once. I don’t know whether I should believe him, because I’ve **never** seen it work. But according to a talk at WinHEC Shenzhen, the WDK for Windows 10 will make this process a reality.

(CONTINUED ON PAGE 7)

Updates to Driver Testing... (Cont.)

(CONTINUED FROM PAGE 6)

The Device Fundamentals, or DevFund, tests have long been a part of the Windows Certification testing matrix. DevFund contains a great set of generic tests for things such as correct PnP, Power, and I/O handling. The Windows 10 WDK reportedly will make it easy to launch the DevFund tests directly from within Visual Studio or from a USB drive, which means that passing DevFund can easily be incorporated into a developer's test procedure. While passing DevFund doesn't exactly mean that your code is bug free, it's a great start to ensuring a baseline of quality in your code.

The Windows 10 WDK brings with it a significant number of improvements, ranging from support for Visual Studio 2015, to support for building Universal Drivers. The ability to easily launch a basic set of tests, with one keystroke, would be the realization of a long-awaited goal. We're looking forward to spending some time testing this feature, and we'll be sure to report back to you what we find.

Follow us!

Yup, Windows Supports That

Qualcomm's DragonBoard 410C Supports Windows

If you follow the goings-on in the world of little single board computers (SBCs), you surely noticed last week's announcement by Qualcomm of the DragonBoard 410C. This ARM-based board includes a 64-bit capable Snapdragon 410 processor, 1GB LPDDR3, and 4GB of eMMC memory. There's an onb-ard MicroSD slot. It has Bluetooth LE, GPS, and WiFi 801.11/a/b/g/n, an HDMI port, and USB 2.0 connectivity.

All this, and it also does Windows

And if that's not enough, there are UART, SPI, two I2C, and twelve available GPIOs. All this in a board that's 96Boards Consumer Edition compliant. That means it's only about 3.5 x 2 inches.

So, again, if you follow such things, you probably heard all that last week. But what you didn't hear last week is that **this little beauty will also run Windows**. This was announced today at WinHEC Shenzhen.

So, now there are two tiny little ARM-based computers that promise Windows support: The Raspberry Pi2, and the DragonBoard 410C. The DragonBoard 410C will be available this summer. I can't wait to see it running Windows Universal Apps and a set of custom-built UMDF 2.15 Universal Drivers.

Follow us!

Tooling up For Windows 10

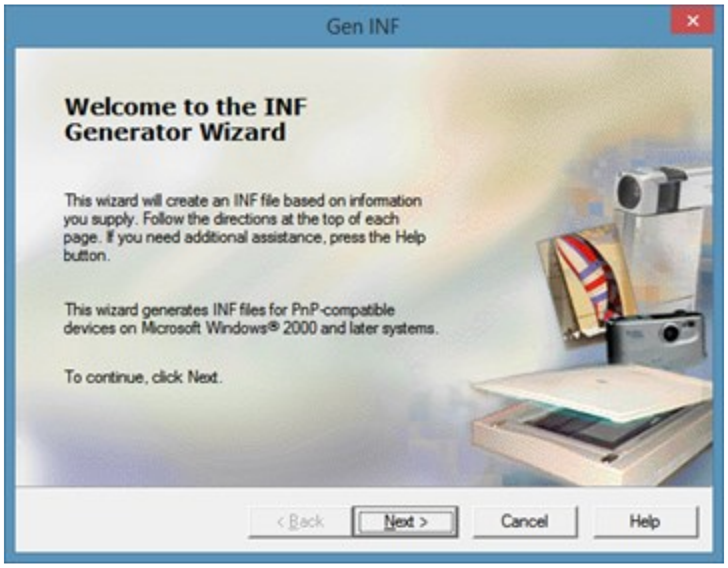
New Tools for Driver Devs Introduced at WinHEC 2015

Across various presentations at WinHEC, several new tools have been introduced for use by Windows driver writers. There's no real overarching theme to these tools, they touch different parts of the driver development process and don't necessarily all target the same audience. However, they're new and potentially useful, so I thought it worth summarizing them here for your convenience.

The INF Validator

It's a bit of a shame that in 2015 we still install our PnP drivers using INFs, which haven't undergone any significant change or improvement since Windows 2000. INFs end up mostly being boilerplate, but they're very prone to error due to their odd rules and syntax. Microsoft has been keenly aware of this problem and attempted to address it over the years with various tools in the WDK.

Remember GenINF, the utility to magically generate INFs? No? Don't worry, it didn't really work anyway and last appeared in the Vista WDK.



Welcome to the INF Generator Wizard

This wizard will create an INF file based on information you supply. Follow the directions at the top of each page. If you need additional assistance, press the Help button.

This wizard generates INF files for PnP-compatible devices on Microsoft Windows® 2000 and later systems.

To continue, click Next.

ChkINF was a far more useful tool that checked your INF for syntax violations. This saved you from typos and places in your INF where you mistakenly had ",,12,,,,1" when you meant to have ",,%12%,,,,1" (duh!).

INFTest was a repackaging of ChkINF, so it had all the same goodness as ChkINF. However, both INFTest and ChkINF had a major downside: even if you *passed* these validators it didn't necessarily guarantee that your driver would install properly.

Enter the INF Validator! Apparently this tool validates your INF by using the *same code*that the installation procedure uses to actually install your INF. Not an interpretation of the code. Not a set of guidelines that the code expects your INF to adhere to. But the same code, just used to validate instead of install. Pretty neat!

The Inflight Trace Recorder

This one is specific to WDF driver writers. All WDF driver writers should be familiar with the WDF Log, which is an in memory trace of the Framework's activity that can be dumped from the debugger with !wdflogdump. Starting in Windows 10, driver writers will be able to add *their own entries* to the WDF Log. Now instead of just seeing Framework activity prior to a system crash you can see driver and Framework activity intermixed.

Additional Windows Performance Analyzer Integration

This one is again specific to WDF driver writers. Here at OSR we love Xperf and the Windows Performance Analyzer. They allow an amazing level of detailed information to be collected in a Windows system for the purpose of performance analysis. The trouble, however, is always in interpreting that data and turning it into actionable changes in your code to resolve performance issues (or squeeze out that next bit of performance).

The next versions of the Frameworks will contain additional trace points to allow for better analysis of performance issues in Framework drivers. Details are scarce on this at the moment, but we'll keep you updated with additional information when we have it.

ACPIGenFx

This one has a pretty narrow focus, but we thought it was too cool to not mention. An ACPI BIOS contains a series of ACPI tables, which are described using the ACPI Source Lanuage (ASL). ASL is compiled into ACPI Machine Language (AML) using the ASL

(CONTINUED ON PAGE 9)

New Tools Introduced... (Cont.)

(CONTINUED FROM PAGE 8)

Compiler (asl.exe) and then interpreted by an OS provided ACPI Machine Language Interpreter (AMLI). The trouble with this is that the AMLI in Windows in not very forgiving. If your ASL is incorrect for whatever reason, you very quickly brick your machine.

The ASL compiler finds syntax errors during the compilation process, but just like any other compiler that doesn't find logical errors in the code. Enter ACPIGenFx, which is a C# library specifically for generating OS independent ASL. Instead of just writing your ASL to text file, you make a series of C# calls to describe the tables you're creating. At build time, ACPIGenFx checks that the resulting ASL "makes sense" and also validates any Windows specific requirements. Assuming that everything checks out, the result of the build is a validated ASL file.

WDK10 and VS2015?

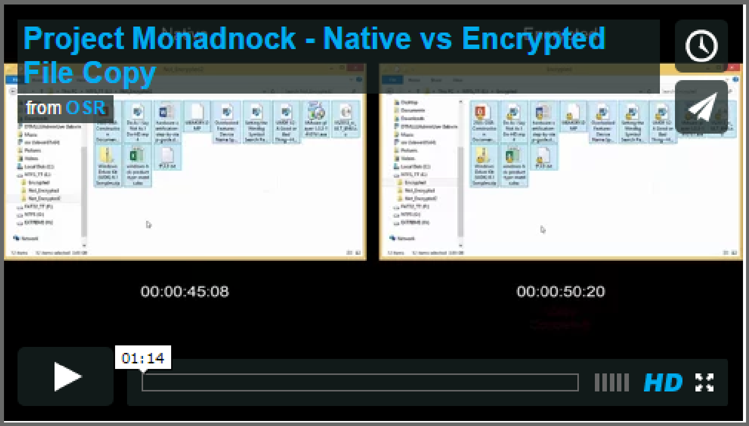
We'll also have a new WDK and Visual Studio version to check out. It's always possible they'll contain something new that wasn't mentioned in the WinHEC presentations. We'll continue to update as details emerge.

Follow us!

FILE ENCRYPTION SOLUTION FRAMEWORK

OSR's next generation solution now available via Early Adopter Program

The OSR File Encryption Solution Framework (formerly "Project Monadnock") allows Clients to incorporate transparent on-access, per-file encryption into their products. While adding on-access encryption sounds like something that should be pretty simple to accomplish, it turns out to be something that's exceptionally complicated. Creating a solution that performs well is even more difficult.



Video demonstrating example copy performance (Tech Preview)

FESF handles most of the necessary complexity, including the actual encryption operations, in kernel mode. This allow Clients that license FESF to build customized file encryption products with no kernel-mode programming.

Early releases of FESF are now available via a limited-access Early Adopter Program (EAP), which provides discounted license terms in exchange for feedback on product development using FESF.

Contact the OSR Sales team at sales@osr.com to learn more.

New Signature Requirements

Microsoft Signatures to be REQUIRED for All Win10 KM Drivers

Nobody likes having to sign their 64-bit Windows kernel-mode drivers. But after you've done it a few times, you get used to it. And after all, you tell yourself, it's probably worth it in terms of security. And at least it's something you can do in-house and Microsoft doesn't have to get involved, right?

Well, that's about to change. According to announcements made today at WinHEC in Shenzhen, kernel-mode code signing as we've come to know it will not be sufficient for your drivers to run on Windows 10.

In order for your driver to be trusted on Windows 10 desktop machines, you will *have* to get a Microsoft signature.

Wait! Stop. Breathe. *This does not mean your drivers will have to pass the Windows Certification tests*.

OK? Have you calmed down? If so, let me explain how this is going to work. For Windows 10 and later, for each kernel-mode driver you want to distribute you will have to:

- Sign your driver package with your company's code signing certificate
- Login to the Microsoft Hardware Developer Portal (AKA sysdev)
- Upload your signed driver package
- Agree to a few particulars
- Download the Microsoft signed files, within minutes.

Of course, you *can* optionally still run your driver through the full battery of tests (using what is referred to today as Compatibility Testing with the Hardware Lab Kit) and submit your logs to sysdev, just as you could in the past. This new option does not require that you pass (or even run!) the tests, and replaces the ability to self-sign drivers.

Why the Change?

For year, security engineers have been saying that driver signing is vulnerable to exploitation. Bad actors have managed to steal certificates, and some have even managed to acquire code signing certificates on their own. It seems that Microsoft agrees.

Starting in Windows 10, to get a Microsoft signature your organization will need to create an account on the Microsoft Hardware Development Portal (http://sysdev.microsoft.com). Creating an account on **sysdev** is both free and easy, but it does require that your organization have a valid code signing certificate. And starting in Windows 10 your organization will need to have an *Extended Validation* Code Signing Certificate.

Extended Validation (EV) certificates require your organization to pass more background checks by the Certification Authority (CA) than ordinary certificates. And they must be stored on "secure hardware tokens." In addition, only CAs that pass an independent audit review can issue EV certificates. Of course, it should go without saying that EV certificates are more expensive than regular Class 3 Code Signing certificates (the lowest cost EV cert we were able to find was $450). Sysdev currently supports EV certificates from Verisign (Symantec) and Digicert.

Therefore, starting in Windows 10, to get any sort of signature from Microsoft (just the driver signing signature or the certification signature) your organization will need to pass the more stringent requirements to qualify for an EV code signing certificate. Microsoft claims this will make driver signing a whole lot safer. However, how this will really play out in the real world depends entirely on how secure the process of obtaining an EV certificate is.

Oh, and one more thing: To obtain the Windows signature you will need to "attest" to the fact that you've comprehensively tested your driver and that you'll monitor the Hardware Developer Portal for issues discovered in your driver and respond to those issues. That seems pretty reasonable to me.

(CONTINUED ON PAGE 11)

Microsoft Signatures... (Cont.)

(CONTINUED FROM PAGE 10)

What's Affected?
These requirements only apply to Windows 10 and later. In fact, Microsoft plans to offer a bit of a grace period: Drivers signed before Windows 10 RTM will be able to use the older signing mechanisms. But once Windows 10 ships, if you want your driver to run on Windows 10 desktop systems, you'll need to (a) get an EV certificate, (b) using that signature submit your driver to sysdev to get Microsoft's signature.

The Ramifications

We don't have the time to fully explore all of the ramifications of this new Windows 10 requirement here in this blog post. We'll delve into this topic further, in future issues of The NT Insider. But we see several interesting complications, including:

- Will smaller, independent, developers and OSS teams be able to obtain the EV certificates necessary to sign their Windows 10 drives?
- Will you need two certificates now? EV Certs require the use of SHA 256. Less than a week ago (when this was written) Microsoft issued a Security Advisory and KB that updates Windows 7 systems to support SHA-256 for code signing. Does this work? We don't know yet. We'll test it, we'll let you know. If it doesn't work, it means you'll have to have *two* Certs for signing your Win7 through Win10 kernel-mode code: One SHA1 cert for signing Win7 drivers and one EV SHA256 for signing everything else, including your submission to sysdev.
- Will the signatures all be additive (in other words, can I sign my components with my SHA1 cert and appropriate cross cert, my EV cert, and then *add* the MSFT signature to those 2), so that I can have three simultaneous signatures at the same time? Will this work properly on older versions of Windows?
- While it's sort of clear what's happening on Windows 10 Client systems, how will these changes impact the next version of Windows Server? So far, we don't know.

Conclusions
Starting in Windows 10, the old way of signing drivers is no longer sufficient. You'll need an EV Code Signing certificate, and you'll need to use the **sysdev** web site to get a Microsoft signature on your driver. As part of getting that signature, you'll need to agree to monitor**s ysdev** for driver problems that are reported and respond to issues. This will only apply on Windows 10 and later, and to drivers built after Windows 10 is released.

If this provides additional real security to the process of creating kernel-mode software, it'll be a good thing. As long as it doesn't simultaneously freeze out small organizations and open software developers.

Stay tuned. We'll be delving into these issues further in the months to come.

Follow us!

OSR Seminar Schedule

Seminar	Dates	Location
Developing File Systems	12-15 May	Boston/Waltham, MA
Internals & Software Drivers	18-22 May	Dulles/Sterling, VA
WDF Drivers: Core Concepts	8-12 June	Boston/Waltham, MA
WDF Drivers: Advanced Implementation Techniques	15-18 June	Boston/Waltham, MA
Kernel Debugging & Crash Analysis	3-7 August	Dulles/Sterling, VA

OSR Seminars
We "Practice What We Teach" For a Reason

When we say "we practice what we teach", this mantra directly translates into the value we bring to our seminars. But don't take our word for it...below are some results from recent surveys of attendees of OSR seminars:

- I've learned everything that I wanted to learn and quite a bit that I did not know I needed.
- I think Scott nicely catered to a diverse audience, some of whom had only taken intro to OS and some who already had experience developing drivers.
- Scott was fantastic. He was very helpful at trying to meet each student's needs. **I will highly recommend him and OSR to my associates.**
- "Peter's **style of teaching is excellent** with the kind of humor and use of objects to give you a visual representation of how things work that was simply amazing.
- "I was very nervous coming in to the seminar as I wasn't sure I had enough hands on experience and background knowledge on Windows internals. The class put my mind at ease and **I was able to quickly grasp and understand the concepts**."
- "I was pleased with the experience. As someone new to driver development, it gave me a better understanding of the framework and **made me a lot more comfortable working with our driver code**."

THE NT INSIDER - Hey...Get Your Own!

Just send a blank email to join-ntinsider@lists.osr.com — and you'll get an email whenever we release a new issue of The NT Insider.

Private Training

A private, on-site seminar format allows you to:

- **Get project specific questions answered.** OSR instructors have the expertise to help your group solve your toughest roadblocks.
- **Customize your seminar.** We know Windows drivers and file systems; take advantage of it. Customize your seminar to fit your group's specific needs.
- **Focus on specific topics.** Spend extra time on topics you really need and less time on topics you already know.
- **Provide an ideal experience.** For groups working on a project or looking to increase their knowledge of a particular topic, OSR's customized on-site seminars are ideal.
- **Save money.** The quote you receive from OSR includes everything you need. There are never additional charges for materials, shipping, or instructor travel.
- **Save more money.** Bringing OSR on-site to teach a seminar costs much less then sending several people to a public class. And you're not paying for your valuable developers to travel.
- **Save time.** Less time out of the office for developers is a good thing.
- **Save hassles.** If you don't have space or lab equipment available, no worries. An OSR seminar consultant can help make arrangements for you.