#### **The AppContainer**

We’ve seen the steps required to create processes back in [Chapter 3](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#ch03); we’ve also seen some of the extra steps required to create UWP processes. The initiation of creation is performed by the DCOMLaunch service, because UWP packages support a set of protocols, one of which is the Launch protocol. The resulting process gets to run inside an AppContainer. Here are several characteristics of packaged processes running inside an AppContainer:

Image The process token integrity level is set to Low, which automatically restricts access to many objects and limits access to certain APIs or functionality for the process, as discussed earlier in this chapter.

Image UWP processes are always created inside a job (one job per UWP app). This job manages the UWP process and any background processes that execute on its behalf (through nested jobs). The jobs allow the Process State Manager (PSM) to suspend or resume the app or background processing in a single stroke.

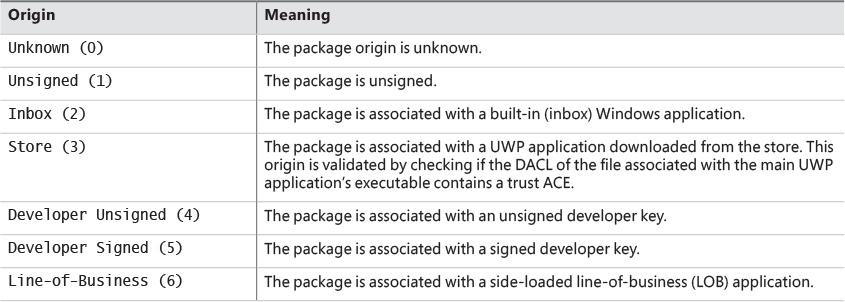
Image The token for UWP processes has an AppContainer SID, which represents a distinct identity based on the SHA-2 hash of the UWP package name. As you’ll see, this SID is used by the system and other applications to explicitly allow access to files and other kernel objects. This SID is part of the APPLICATION PACKAGE AUTHORITY instead of the NT AUTHORITY you’ve mostly seen so far in this chapter. Thus, it begins with S-1-15-2 in its string format, corresponding to SECURITY\_APP\_PACKAGE\_BASE\_RID (15) and SECURITY\_APP\_PACKAGE\_BASE\_RID (2). Because a SHA-2 hash is 32 bytes, there are a total of eight RIDs (recall that a RID is the size of a 4-byte ULONG) in the remainder of the SID.

Image The token may contain a set of capabilities, each represented with a SID. These capabilities are declared in the application manifest and shown on the app’s page in the store. Stored in the capability section of the manifest, they are converted to SID format using rules we’ll see shortly, and belong to the same SID authority as in the previous bullet, but using the well-known SECURITY\_CAPABILITY\_BASE\_RID (3) instead. Various components in the Windows Runtime, user-mode device-access classes, and kernel can look for capabilities to allow or deny certain operations.

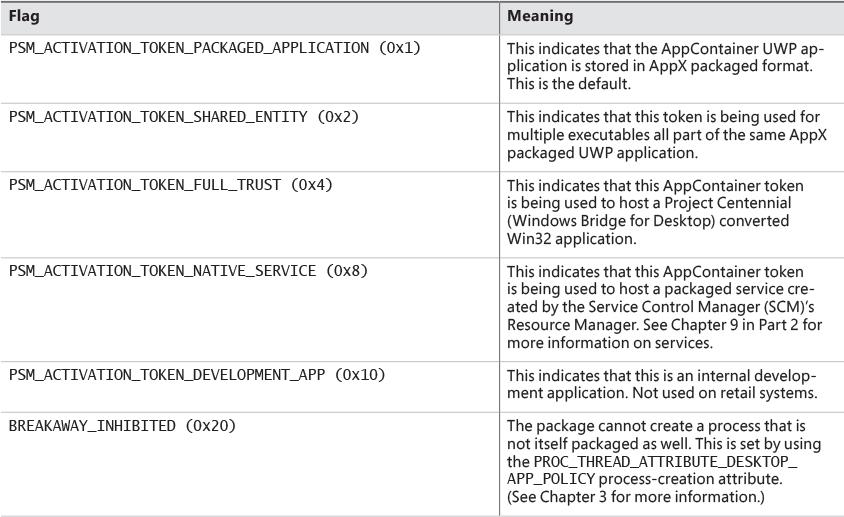
Image The token may only contain the following privileges: SeChangeNotifyPrivilege, SeIncrease-WorkingSetPrivilege, SeShutdownPrivilege, SeTimeZonePrivilege, and SeUndockPrivilege. These are the default set of privileges associated with standard user accounts. Additionally, the AppContainerPrivilegesEnabledExt function part of the ms-win-ntos-ksecurity API Set contract extension can be present on certain devices to further restrict which privileges are enabled by default.

Image The token will contain up to four security attributes (see the section on attribute-based access control earlier in this chapter) that identify this token as being associated with a UWP packaged application. These attributes are added by the DcomLaunch service as indicated earlier, which is responsible for the activation of UWP applications. They are as follows:

• **WIN://PKG** This identifies this token as belonging to a UWP packaged application. It contains an integer value with the application’s origin as well as some flags. See [Table 7-13](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07tab13) and [Table 7-14](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07tab14) for these values.



**TABLE 7-13** Package origins



**TABLE 7-14** Package flags

• **WIN://SYSAPPID** This contains the application identifiers (called *package monikers* or *string names*) as an array of Unicode string values.

• **WIN://PKGHOSTID** This identifies the UWP package host ID for packages that have an explicit host through an integer value.

• **WIN://BGKD** This is only used for background hosts (such as the generic background task host BackgroundTaskHost.exe) that can store packaged UWP services running as COM providers. The attribute’s name stands for *background* and contains an integer value that stores its explicit host ID.

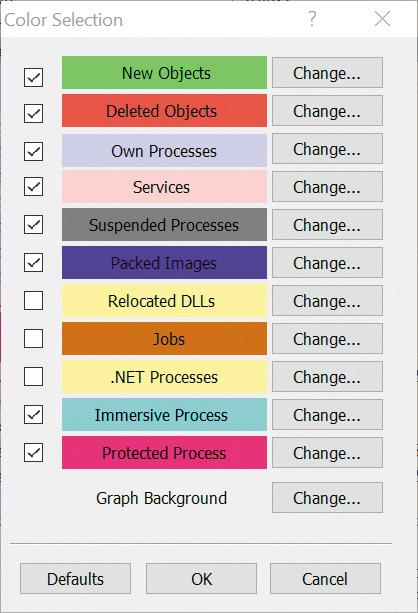
The TOKEN\_LOWBOX (0x4000) flag will be set in the token’s Flags member, which can be queried with various Windows and kernel APIs (such as GetTokenInformation). This allows components to identity and operate differently under the presence of an AppContainer token.

Image Note

A second type of AppContainer exists: a child AppContainer. This is used when a UWP AppContainer (or parent AppContainer) wishes to create its own nested AppContainer to further lock down the security of the application. Instead of eight RIDs, a child AppContainer has four additional RIDs (the first eight match the parents’) to uniquely identify it.

EXPERIMENT: Viewing UWP process information

There are several ways to look at UWP processes, some more obvious than others. Process Explorer can highlight processes that use the Windows Runtime in color (cyan by default). To see this in action, open Process Explorer, open the **Options** menu, and choose **Configure Colors**. Then make sure the **Immersive Processes** check box is selected.



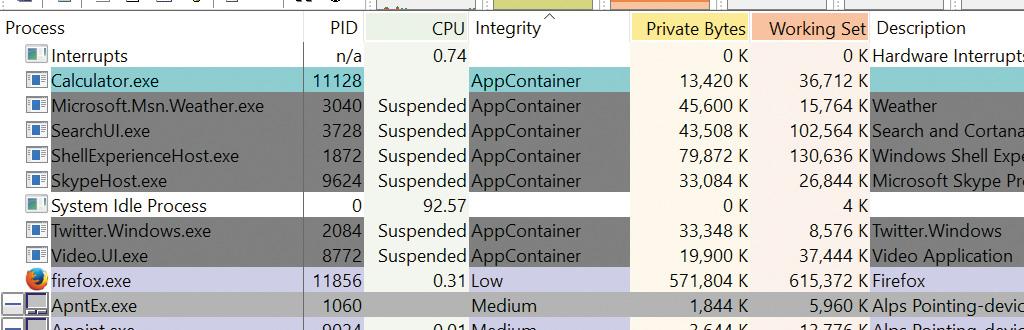
*Immersive process* is the original term used to describe WinRT (now UWP) apps in Windows 8. (They were mostly full screen, and therefore “immersive.”) This distinction is available by calling the IsImmersiveProcess API.

Run Calc.exe and switch to Process Explorer. You should see several processes highlighted in cyan, including Calculator.exe. Now minimize the Calculator app and notice that the cyan highlight has turned gray. This is because Calculator has been suspended. Restore Calculator’s window, and it’s back to cyan.

You should have similar experiences with other apps—for example, Cortana (SearchUI.exe). Click or tap the **Cortana** icon on the taskbar and then close it. You should see the gray to cyan and back to gray transition. Or, click or tap the **Start** button. ShellExperienceHost.exe highlights in a similar fashion.

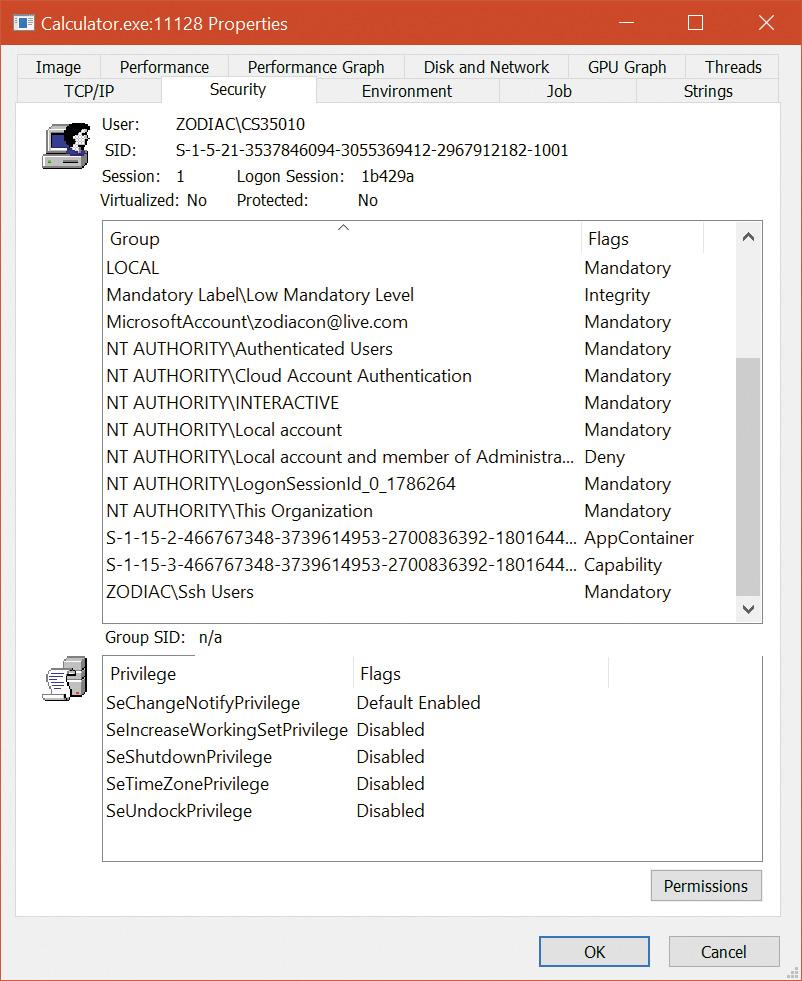
The presence of some cyan-highlighted processes might surprise you, such as Explorer.exe, TaskMgr.Exe, and RuntimeBroker.exe. These are not really apps, but use Windows Runtime APIs, and so are classified as immersive. (The role of RuntimeBroker will be discussed shortly.)

Finally, make sure the Integrity column is visible in Process Explorer and sort by that column. You’ll find processes such as Calculator.exe and SearchUI.exe with AppContainer integrity level. Notice that Explorer and TaskMgr are not there, clearly showing they are not UWP processes, and so live under different rules.



EXPERIMENT: Viewing an AppContainer token

You can look at the properties of an AppContainer hosted process with several tools. In Process Explorer, the Security tab shows the capabilities associated with the token. Here’s the Security tab for Calculator.exe:



Notice two interesting pieces of information: the AppContainer SID, shown in the Flags column as AppContainer, and a single capability, right underneath the AppContainer SID. Except for the base RID (SECURITY\_APP\_PACKAGE\_BASE\_RID versus SECURITY\_CAPABILITY\_BASE\_RID), the remaining eight RIDs are identical, and both refer to the package name in SHA-2 format as discussed. This shows you that there will always be one implicit capability, the capability of being the package itself, which really means Calculator requires no capabilities at all. The upcoming capabilities section covers a much more complex example.

EXPERIMENT: Viewing AppContainer token attributes

You can obtain similar information on the command line by using the AccessChk Sysinternals tool while also adding a full list of all of the token’s attributes. For example, running AccessChk with the -p -f switches followed by the process ID for SearchUI.exe, which hosts Cortana, shows the following:

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p691pro01a)

C:\ >accesschk -p -f 3728

Accesschk v6.10 - Reports effective permissions for securable objects

Copyright (C) 2006-2016 Mark Russinovich

Sysinternals - www.sysinternals.com

[7416] SearchUI.exe

RW DESKTOP-DD6KTPM\aione

RW NT AUTHORITY\SYSTEM

RW Package\S-1-15-2-1861897761-1695161497-2927542615-642690995-327840285-2659745135-2630312742

Token security:

RW DESKTOP-DD6KTPM\aione

RW NT AUTHORITY\SYSTEM

RW DESKTOP-DD6KTPM\aione-S-1-5-5-0-459087

RW Package\S-1-15-2-1861897761-1695161497-2927542615-642690995-327840285-2659745135-2630312742

R BUILTIN\Administrators

Token contents:

User:

DESKTOP-DD6KTPM\aione

AppContainer:

Package

\S-1-15-2-1861897761-1695161497-2927542615-642690995-327840285-2659745135-2630312742

Groups:

Mandatory Label\Low Mandatory Level INTEGRITY

Everyone MANDATORY

NT AUTHORITY\Local account and member of Administrators group DENY

...

Security Attributes:

WIN://PKGHOSTID

TOKEN\_SECURITY\_ATTRIBUTE\_TYPE\_UINT64

[0] 1794402976530433

WIN://SYSAPPID

TOKEN\_SECURITY\_ATTRIBUTE\_TYPE\_STRING

[0] Microsoft.Windows.Cortana\_1.8.3.14986\_neutral\_neutral\_cw5n1h2txyewy

[1] CortanaUI

[2] Microsoft.Windows.Cortana\_cw5n1h2txyewy

WIN://PKG

TOKEN\_SECURITY\_ATTRIBUTE\_TYPE\_UINT64

[0] 131073

TSA://ProcUnique

[TOKEN\_SECURITY\_ATTRIBUTE\_NON\_INHERITABLE]

[TOKEN\_SECURITY\_ATTRIBUTE\_COMPARE\_IGNORE]

TOKEN\_SECURITY\_ATTRIBUTE\_TYPE\_UINT64

[0] 204

[1] 24566825

First is the package host ID, converted to hex: 0x6600000000001. Because all package host IDs begin with 0x66, this means Cortana is using the first available host identifier: 1. Next are the system application IDs, which contain three strings: the strong package moniker, the friendly application name, and the simplified package name. Finally, you have the package claim, which is 0x20001 in hex. Based on the [Table 7-13](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07tab13) and [Table 7-14](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07tab14) fields you saw, this indicates an origin of Inbox (2) and flags set to PSM\_ACTIVATION\_TOKEN\_PACKAGED\_APPLICATION, confirming that Cortana is part of an AppX package.

##### **AppContainer security environment**

One of the biggest side-effects caused by the presence of an AppContainer SID and related flags is that the access check algorithm you saw in the “[Access checks](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#ch07lev2sec5)” section earlier in this chapter is modified to essentially ignore all regular user and group SIDs that the token may contain, essentially treating them as deny-only SIDs. This means that even though Calculator may be launched by a user John Doe belonging to the Users and Everyone groups, it will fail any access checks that grant access to John Doe’s SID, the Users group SID, or the Everyone group SID. In fact, the only SIDs that are checked during the discretionary access check algorithm will be that of the AppContainer SID, followed by the capability access check algorithm, which will look at any capability SIDs part of the token.

Taking things even further than merely treating the discretionary SIDs as deny-only, AppContainer tokens effect one further critical security change to the access check algorithm: a NULL DACL, typically treated as an allow-anyone situation due to the lack of any information (recall that this is different from an empty DACL, which is a deny-everyone situation due to explicit allow rules), is ignored and treated as a deny situation. To make matters simple, the only types of securable objects that an AppContainer can access are those that explicitly have an allow ACE for its AppContainer SID or for one of its capabilities. Even unsecured (NULL DACL) objects are out of the game.

This situation causes compatibility problems. Without access to even the most basic file system, registry, and object manager resources, how can an application even function? Windows takes this into account by preparing a custom execution environment, or “jail” if you will, specifically for each AppContainer. These jails are as follows:

Image Note

So far we’ve implied that each UWP packaged application corresponds to one AppContainer token. However, this doesn’t necessarily imply that only a single executable file can be associated with an AppContainer. UWP packages can contain multiple executable files, which all belong to the same AppContainer. This allows them to share the same SID and capabilities and exchange data between each other, such as a micro-service back-end executable and a foreground front-end executable.

Image The AppContainer SID’s string representation is used to create a subdirectory in the object manager’s namespace under \Sessions\x\AppContainerNamedObjects. This becomes the private directory of named kernel objects. This specific subdirectory object is then ACLed with the AppContainer SID associated with the AppContainer that has an allow-all access mask. This is in contrast to desktop apps, which all use the \Sessions\x\BaseNamedObjects subdirectory (within the same session x). We’ll discuss the implications of that shortly, as well as the requirement for the token to now store handles.

Image The token will contain a LowBox number, which is a unique identifier into an array of LowBox Number Entry structures that the kernel stores in the g\_SessionLowboxArray global variable. Each of these maps to a SEP\_LOWBOX\_NUMBER\_ENTRY structure that, most importantly, contains an atom table unique to this AppContainer, because the Windows Subsystem Kernel Mode Driver (Win32k.sys) does not allow AppContainers access to the global atom table.

Image The file system contains a directory in %LOCALAPPDATA% called Packages. Inside it are the package monikers (the string version of the AppContainer SID—that is, the package name) of all the installed UWP applications. Each of these application directories contains application-specific directories, such as TempState, RoamingState, Settings, LocalCache, and others, which are all ACLed with the specific AppContainer SID corresponding to the application, set to an allow-all access mask.

Image Within the Settings directory is a Settings.dat file, which is a registry hive file that is loaded as an application hive. (You will learn more about application hives in Chapter 9 in Part 2.) The hive acts as the local registry for the application, where WinRT APIs store the various persistent state of the application. Once again, the ACL on the registry keys explicitly grants allow-all access to the associated AppContainer SID.

These four jails allow AppContainers to securely, and locally, store their file system, registry, and atom table without requiring access to sensitive user and system areas on the system. That being said, what about the ability to access, at least in read-only mode, critical system files (such as Ntdll.dll and Kernel32.dll) or registry keys (such as the ones these libraries will need), or even named objects (such as the \RPC Control\DNSResolver ALPC port used for DNS lookups)? It would not make sense, on each UWP application or uninstallation, to re-ACL entire directories, registry keys, and object namespaces to add or remove various SIDs.

To solve this problem, the security subsystem understands a specific group SID called ALL APPLICATION PACKAGES, which automatically binds itself to any AppContainer token. Many critical system locations, such as %SystemRoot%\System32 and HKLM\Software\Microsoft\Windows\CurrentVersion, will have this SID as part of their DACL, typically with a read or read-and-execute access mask. Certain objects in the object manager namespace will have this as well, such as the DNSResolver ALPC port in the \RPC Control object manager directory. Other examples include certain COM objects, which grant the execute right. Although not officially documented, third-party developers, as they create non-UWP applications, can also allow interactions with UWP applications by also applying this SID to their own resources.

Unfortunately, because UWP applications can technically load almost any Win32 DLL as part of their WinRT needs (because WinRT is built on top of Win32, as you saw), and because it’s hard to predict what an individual UWP application might need, many system resources have the ALL APPLICATION PACKAGES SID associated with their DACL as a precaution. This now means there is no way for a UWP developer, for example, to prevent DNS lookups from their application. This greater-than-needed access is also helpful for exploit writers, which could leverage it to escape from the AppContainer sandbox. Newer versions of Windows 10, starting with version 1607 (Anniversary Update), contain an additional element of security to combat this risk: Restricted AppContainers.

By using the PROC\_THREAD\_ATTRIBUTE\_ALL\_APPLICATION\_PACKAGES\_POLICY process attribute and setting it to PROCESS\_CREATION\_ALL\_APPLICATION\_PACKAGES\_OPT\_OUT during process creation (see [Chapter 3](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#ch03) for more information on process attributes), the token will not be associated with any ACEs that specify the ALL APPLICATION PACKAGES SID, cutting off access to many system resources that would otherwise be accessible. Such tokens can be identified by the presence of a fourth token attribute named WIN://NOALLAPPPKG with an integer value set to 1.

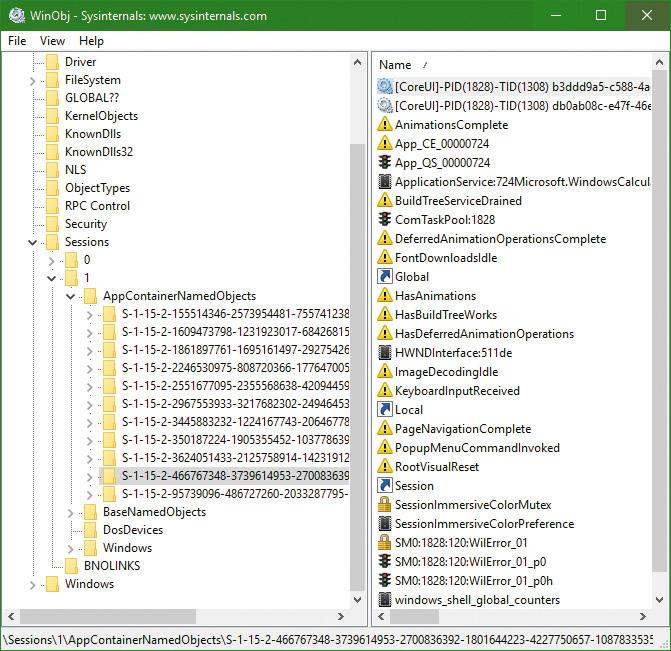
Of course, this takes us back to the same problem: How would such an application even be able to load Ntdll.dll, which is key to any process initialization? Windows 10 version 1607 introduces a new group, called ALL RESTRICTED APPLICATION PACKAGES, which takes care of this problem. For example, the System32 directory now also contains this SID, also set to allow read and execute permissions, because loading DLLs in this directory is key even to the most sandboxed process. However, the DNSResolver ALPC port does not, so such an AppContainer would lose access to DNS.

EXPERIMENT: Viewing AppContainer security attributes

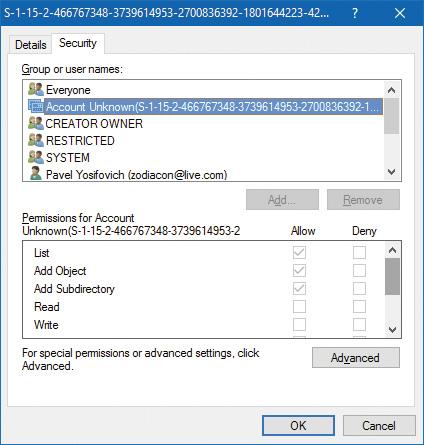
In this experiment, we’ll look at the security attributes of some of the directories mentioned in the previous section.

**1.** Make sure Calculator is running.

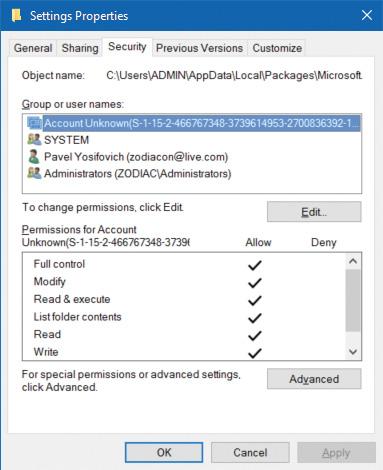
**2.** Open WinObj elevated from Sysinternals and navigate to the object directory corresponding to Calculator’s AppContainer SID. (You saw it in a previous experiment.)



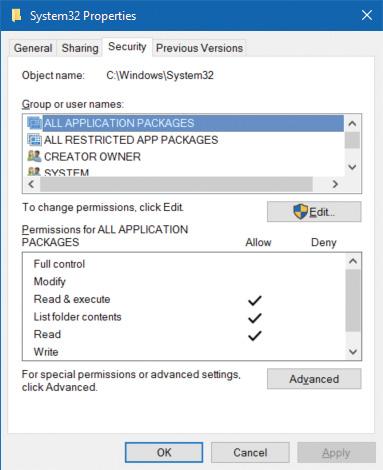
**3.** Right-click the directory, select **Properties**, and click the **Security** tab. You should see something like the following screenshot. Calculator’s AppContainer SID has permission to list, add object, and add subdirectory (among others scrolled out of view), which simply means Calculator can create kernel objects under this directory.



**4.** Open Calculator’s local folder by navigating to %LOCALAPPDATA%\Packages\Microsoft.WindowsCalculator\_8wekyb3d8bbwe. Then right-click the **Settings** subdirectory, select **Properties**, and click the **Security** tab. You should see Calculator’s AppContainer SID having full permissions for the folder:



**5.** In Explorer, open the %SystemRoot% directory (for example, C:\Windows), right-click the **System32** directory, select **Properties,** and click the **Security** tab. You should see the read and execute permissions for all application packages and all restricted application packages (if using Windows 10 version 1607 or later):



As an alternative, you can use the AccessChk Sysinternals command-line tool to view the same information.

EXPERIMENT: Viewing the AppContainer atom table

An atom table is a hash table of integers to strings that’s used by the windowing system for various identification purposes, such as Window Class registration (RegisterClassEx) and custom Windows messages. The AppContainer private atom table can be viewed with the kernel debugger:

**1.** Run Calculator, open WinDbg, and start local kernel debugging.

**2.** Find the Calculator process:

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p697pro01a)

lkd> !process 0 1 calculator.exe

PROCESS ffff828cc9ed1080

SessionId: 1 Cid: 4bd8 Peb: d040bbc000 ParentCid: 03a4

DeepFreeze

DirBase: 5fccaa000 ObjectTable: ffff950ad9fa2800 HandleCount:

<Data Not Accessible>

Image: Calculator.exe

VadRoot ffff828cd2c9b6a0 Vads 168 Clone 0 Private 2938. Modified 3332.

Locked 0.

DeviceMap ffff950aad2cd2f0

Token **ffff950adb313060**

...

**3.** Use the token value with the following expressions:

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p697pro02a)

lkd> r? @$t1 = @$t0->NumberOfBuckets

lkd> r? @$t0 = (nt!\_RTL\_ATOM\_TABLE\*)((nt!\_token\*)**0xffff950adb313060**)-

>LowboxNumberEntry->AtomTable

lkd> .for (r @$t3 = 0; @$t3 < @$t1; r @$t3 = @$t3 + 1) { ?? (wchar\_t\*)@$t0-

>Buckets[@$t3]->Name }

wchar\_t \* 0xffff950a'ac39b78a

"Protocols"

wchar\_t \* 0xffff950a'ac17b7aa

"Topics"

wchar\_t \* 0xffff950a'b2fd282a

"TaskbarDPI\_Deskband"

wchar\_t \* 0xffff950a'b3e2b47a

"Static"

wchar\_t \* 0xffff950a'b3c9458a

"SysTreeView32"

wchar\_t \* 0xffff950a'ac34143a

"UxSubclassInfo"

wchar\_t \* 0xffff950a'ac5520fa

"StdShowItem"

wchar\_t \* 0xffff950a'abc6762a

"SysSetRedraw"

wchar\_t \* 0xffff950a'b4a5340a

"UIA\_WindowVisibilityOverridden"

wchar\_t \* 0xffff950a'ab2c536a

"True"

...

wchar\_t \* 0xffff950a'b492c3ea

"tooltips\_class"

wchar\_t \* 0xffff950a'ac23f46a

"Save"

wchar\_t \* 0xffff950a'ac29568a

"MSDraw"

wchar\_t \* 0xffff950a'ac54f32a

"StdNewDocument"

wchar\_t \* 0xffff950a'b546127a

"{FB2E3E59-B442-4B5B-9128-2319BF8DE3B0}"

wchar\_t \* 0xffff950a'ac2e6f4a

"Status"

wchar\_t \* 0xffff950a'ad9426da

"ThemePropScrollBarCtl"

wchar\_t \* 0xffff950a'b3edf5ba

"Edit"

wchar\_t \* 0xffff950a'ab02e32a

"System"

wchar\_t \* 0xffff950a'b3e6c53a

"MDIClient"

wchar\_t \* 0xffff950a'ac17a6ca

"StdDocumentName"

wchar\_t \* 0xffff950a'ac6cbeea

"StdExit"

wchar\_t \* 0xffff950a'b033c70a

"{C56C5799-4BB3-7FAE-7FAD-4DB2F6A53EFF}"

wchar\_t \* 0xffff950a'ab0360fa

"MicrosoftTabletPenServiceProperty"

wchar\_t \* 0xffff950a'ac2f8fea

"OLEsystem"

##### **AppContainer capabilities**

As you’ve just seen, UWP applications have very restricted access rights. So how, for example, is the Microsoft Edge application able to parse the local file system and open PDF files in the user’s Documents folder? Similarly, how can the Music application play MP3 files from the Music directory? Whether done directly through kernel access checks or by brokers (which you’ll see in the next section), the key lies in capability SIDs. Let’s see where these come from, how they are created, and when they are used.

First, UWP developers begin by creating an application manifest that specifies many details of their application, such as the package name, logo, resources, supported devices, and more. One of the key elements for capability management is the list of capabilities in the manifest. For example, let’s take a look at Cortana’s application manifest, located in %SystemRoot%\SystemApps\Microsoft.Windows.Cortana\_cw5n1h2txywey\AppxManifest.xml:

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p699pro01a)

<Capabilities>

<wincap:Capability Name="packageContents"/>

<!-- Needed for resolving MRT strings -->

<wincap:Capability Name="cortanaSettings"/>

<wincap:Capability Name="cloudStore"/>

<wincap:Capability Name="visualElementsSystem"/>

<wincap:Capability Name="perceptionSystem"/>

<Capability Name="internetClient"/>

<Capability Name="internetClientServer"/>

<Capability Name="privateNetworkClientServer"/>

<uap:Capability Name="enterpriseAuthentication"/>

<uap:Capability Name="musicLibrary"/>

<uap:Capability Name="phoneCall"/>

<uap:Capability Name="picturesLibrary"/>

<uap:Capability Name="sharedUserCertificates"/>

<rescap:Capability Name="locationHistory"/>

<rescap:Capability Name="userDataSystem"/>

<rescap:Capability Name="contactsSystem"/>

<rescap:Capability Name="phoneCallHistorySystem"/>

<rescap:Capability Name="appointmentsSystem"/>

<rescap:Capability Name="chatSystem"/>

<rescap:Capability Name="smsSend"/>

<rescap:Capability Name="emailSystem"/>

<rescap:Capability Name="packageQuery"/>

<rescap:Capability Name="slapiQueryLicenseValue"/>

<rescap:Capability Name="secondaryAuthenticationFactor"/>

<DeviceCapability Name="microphone"/>

<DeviceCapability Name="location"/>

<DeviceCapability Name="wiFiControl"/>

</Capabilities>

You’ll see many types of entries in this list. For example, the Capability entries contain the well-known SIDs associated with the original capability set that was implemented in Windows 8. These begin with SECURITY\_CAPABILITY\_—for example, SECURITY\_CAPABILITY\_INTERNET\_CLIENT, which is part of the capability RID under the APPLICATION PACKAGE AUTHORITY. This gives us a SID of S-1-15-3-1 in string format.

Other entries are prefixed with uap, rescap, and wincap. One of these (rescap) refers to restricted capabilities. These are capabilities that require special onboarding from Microsoft and custom approvals before being allowed on the store. In Cortana’s case, these include capabilities such as accessing SMS text messages, emails, contacts, location, and user data. Windows capabilities, on the other hand, refer to capabilities that are reserved for Windows and system applications. No store application can use these. Finally, UAP capabilities refer to standard capabilities that anyone can request on the store. (Recall that UAP is the older name for UWP.)

Unlike the first set of capabilities, which map to hard-coded RIDs, these capabilities are implemented in a different fashion. This ensures a list of well-known RIDs doesn’t have to be constantly maintained. Instead, with this mode, capabilities can be fully custom and updated on the fly. To do this, they simply take the capability string, convert it to full upper-case format, and take a SHA-2 hash of the resulting string, much like AppContainer package SIDs are the SHA-2 hash of the package moniker. Again, since SHA-2 hashes are 32 bytes, this results in 8 RIDs for each capability, following the well-known SECURITY\_CAPABILITY\_BASE\_RID (3).

Finally, you’ll notice a few DeviceCapability entries. These refer to device classes that the UWP application will need to access, and can be identified either through well-known strings such as the ones you see above or directly by a GUID that identifies the device class. Rather than using one of the two methods of SID creation already described, this one uses yet a third! For these types of capabilities, the GUID is converted into a binary format and then mapped out into four RIDs (because a GUID is 16 bytes). On the other hand, if a well-known name was specified instead, it must first be converted to a GUID. This is done by looking at the HKLM\Software\Microsoft\Windows\CurrentVersion\DeviceAccess\CapabilityMappings registry key, which contains a list of registry keys associated with device capabilities and a list of GUIDs that map to these capabilities. The GUIDs are then converted to a SID as you’ve just seen.

Image Note

For an up-to-date list of supported capabilities, see <https://msdn.microsoft.com/en-us/windows/uwp/packaging/app-capability-declarations>.

As part of encoding all of these capabilities into the token, two additional rules are applied:

Image As you may have seen in the earlier experiment, each AppContainer token contains its own package SID encoded as a capability. This can be used by the capability system to specifically lock down access to a particular app through a common security check instead of obtaining and validating the package SID separately.

Image Each capability is re-encoded as a group SID through the use of the SECURITY\_CAPABILITY\_APP\_RID (1024) RID as an additional sub-authority preceding the regular eight-capability hash RIDs.

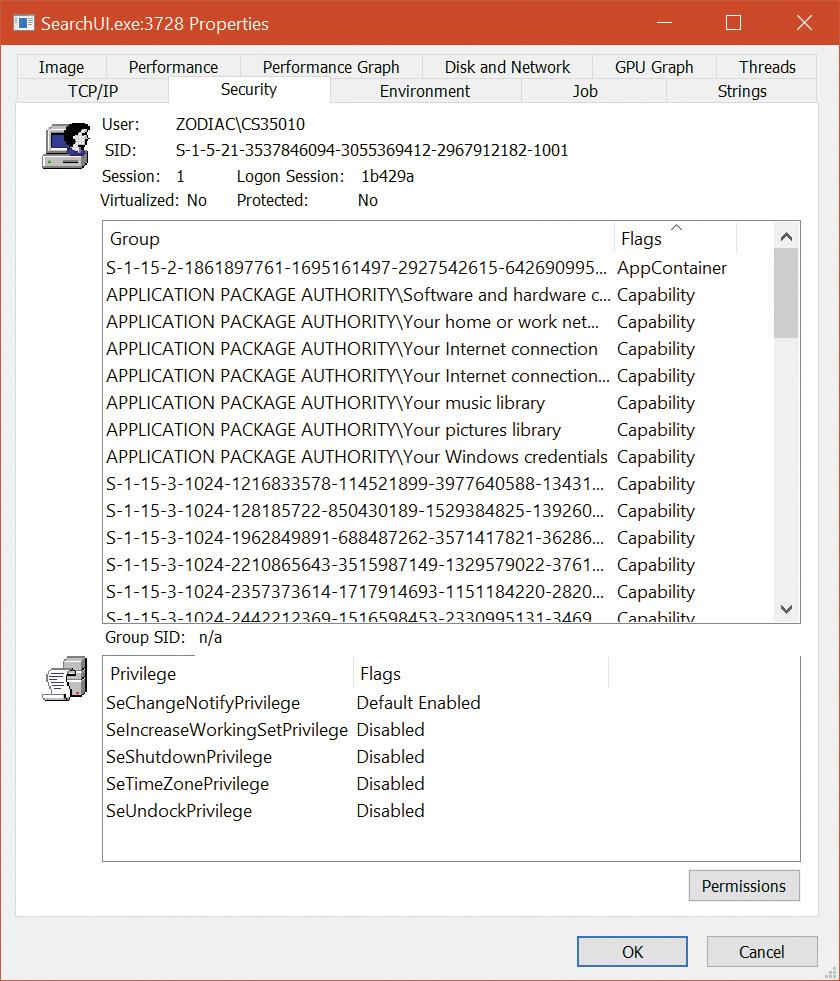
After the capabilities are encoded into the token, various components of the system will read them to determine whether an operation being performed by an AppContainer should be permitted. You’ll note most of the APIs are undocumented, as communication and interoperability with UWP applications is not officially supported and best left to broker services, inbox drivers, or kernel components. For example, the kernel and drivers can use the RtlCapabilityCheck API to authenticate access to certain hardware interfaces or APIs.

As an example, the Power Manager checks for the ID\_CAP\_SCREENOFF capability before allowing a request to shut off the screen from an AppContainer. The Bluetooth port driver checks for the bluetoothDiagnostics capability, while the application identity driver checks for Enterprise Data Protection (EDP) support through the enterpriseDataPolicy capability. In user mode, the documented CheckTokenCapability API can be used, although it must know the capability SID instead of providing the name (the undocumented RtlDeriveCapabilitySidFromName can generate this, however). Another option is the undocumented CapabilityCheck API, which does accept a string.

Finally, many RPC services leverage the RpcClientCapabilityCheck API, which is a helper function that takes care of retrieving the token and requires only the capability string. This function is very commonly used by many of the WinRT-enlightened services and brokers, which utilize RPC to communicate with UWP client applications.

EXPERIMENT: Viewing AppContainer capabilities

To clearly demonstrate all these various capability combinations and their population in the token, let’s look at the capabilities for a complex app such as Cortana. You’ve already seen its manifest, so you can use that output to compare with the UI. First, looking at the Security tab for SearchUI.exe shows the following (sorted by the Flags column):



Clearly, Cortana has obtained many capabilities—all the ones in its manifest. Some are those that were originally in Windows 8 and are known to functions like IsWellKnownSid, for which Process Explorer shows a friendly name. Other capabilities are just shown using their SID, as they represent either hashes or GUIDs, as discussed.

To get the details of the package from which the UWP process was created, you can use the UWPList tool provided with the downloadable resources for this book. It can show all immersive processes on the system or a single process based on its ID:

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p702pro01a)

C:\WindowsInternals>UwpList.exe 3728

List UWP Processes - version 1.1 (C)2016 by Pavel Yosifovich

Building capabilities map... done.

Process ID: 3728

------------------

Image name: C:\Windows\SystemApps\Microsoft.Windows.Cortana\_cw5n1h2txyewy\SearchUI.exe

Package name: Microsoft.Windows.Cortana

Publisher: CN=Microsoft Windows, O=Microsoft Corporation, L=Redmond, S=Washington, C=US

Published ID: cw5n1h2txyewy

Architecture: Neutral

Version: 1.7.0.14393

AppContainer SID: S-1-15-2-1861897761-1695161497-2927542615-642690995-327840285-

2659745135-2630312742

Lowbox Number: 3

Capabilities: 32

cortanaSettings (S-1-15-3-1024-1216833578-114521899-3977640588-1343180512-

2505059295-473916851-3379430393-3088591068) (ENABLED)

visualElementsSystem (S-1-15-3-1024-3299255270-1847605585-2201808924-710406709-

3613095291-873286183-3101090833-2655911836) (ENABLED)

perceptionSystem (S-1-15-3-1024-34359262-2669769421-2130994847-3068338639-

3284271446-2009814230-2411358368-814686995) (ENABLED)

internetClient (S-1-15-3-1) (ENABLED)

internetClientServer (S-1-15-3-2) (ENABLED)

privateNetworkClientServer (S-1-15-3-3) (ENABLED)

enterpriseAuthentication (S-1-15-3-8) (ENABLED)

musicLibrary (S-1-15-3-6) (ENABLED)

phoneCall (S-1-15-3-1024-383293015-3350740429-1839969850-1819881064-1569454686-

4198502490-78857879-1413643331) (ENABLED)

picturesLibrary (S-1-15-3-4) (ENABLED)

sharedUserCertificates (S-1-15-3-9) (ENABLED)

locationHistory (S-1-15-3-1024-3029335854-3332959268-2610968494-1944663922-

1108717379-267808753-1292335239-2860040626) (ENABLED)

userDataSystem (S-1-15-3-1024-3324773698-3647103388-1207114580-2173246572-

4287945184-2279574858-157813651-603457015) (ENABLED)

contactsSystem (S-1-15-3-1024-2897291008-3029319760-3330334796-465641623-3782203132-

742823505-3649274736-3650177846) (ENABLED)

phoneCallHistorySystem (S-1-15-3-1024-2442212369-1516598453-2330995131-3469896071-

605735848-2536580394-3691267241-2105387825) (ENABLED)

appointmentsSystem (S-1-15-3-1024-2643354558-482754284-283940418-2629559125-

2595130947-547758827-818480453-1102480765) (ENABLED)

chatSystem (S-1-15-3-1024-2210865643-3515987149-1329579022-3761842879-3142652231-

371911945-4180581417-4284864962) (ENABLED)

smsSend (S-1-15-3-1024-128185722-850430189-1529384825-139260854-329499951-

1660931883-3499805589-3019957964) (ENABLED)

emailSystem (S-1-15-3-1024-2357373614-1717914693-1151184220-2820539834-3900626439-

4045196508-2174624583-3459390060) (ENABLED)

packageQuery (S-1-15-3-1024-1962849891-688487262-3571417821-3628679630-802580238-

1922556387-206211640-3335523193) (ENABLED)

slapiQueryLicenseValue (S-1-15-3-1024-3578703928-3742718786-7859573-1930844942-

2949799617-2910175080-1780299064-4145191454) (ENABLED)

S-1-15-3-1861897761-1695161497-2927542615-642690995-327840285-2659745135-2630312742

(ENABLED)

S-1-15-3-787448254-1207972858-3558633622-1059886964 (ENABLED)

S-1-15-3-3215430884-1339816292-89257616-1145831019 (ENABLED)

S-1-15-3-3071617654-1314403908-1117750160-3581451107 (ENABLED)

S-1-15-3-593192589-1214558892-284007604-3553228420 (ENABLED)

S-1-15-3-3870101518-1154309966-1696731070-4111764952 (ENABLED)

S-1-15-3-2105443330-1210154068-4021178019-2481794518 (ENABLED)

S-1-15-3-2345035983-1170044712-735049875-2883010875 (ENABLED)

S-1-15-3-3633849274-1266774400-1199443125-2736873758 (ENABLED)

S-1-15-3-2569730672-1095266119-53537203-1209375796 (ENABLED)

S-1-15-3-2452736844-1257488215-2818397580-3305426111 (ENABLED)

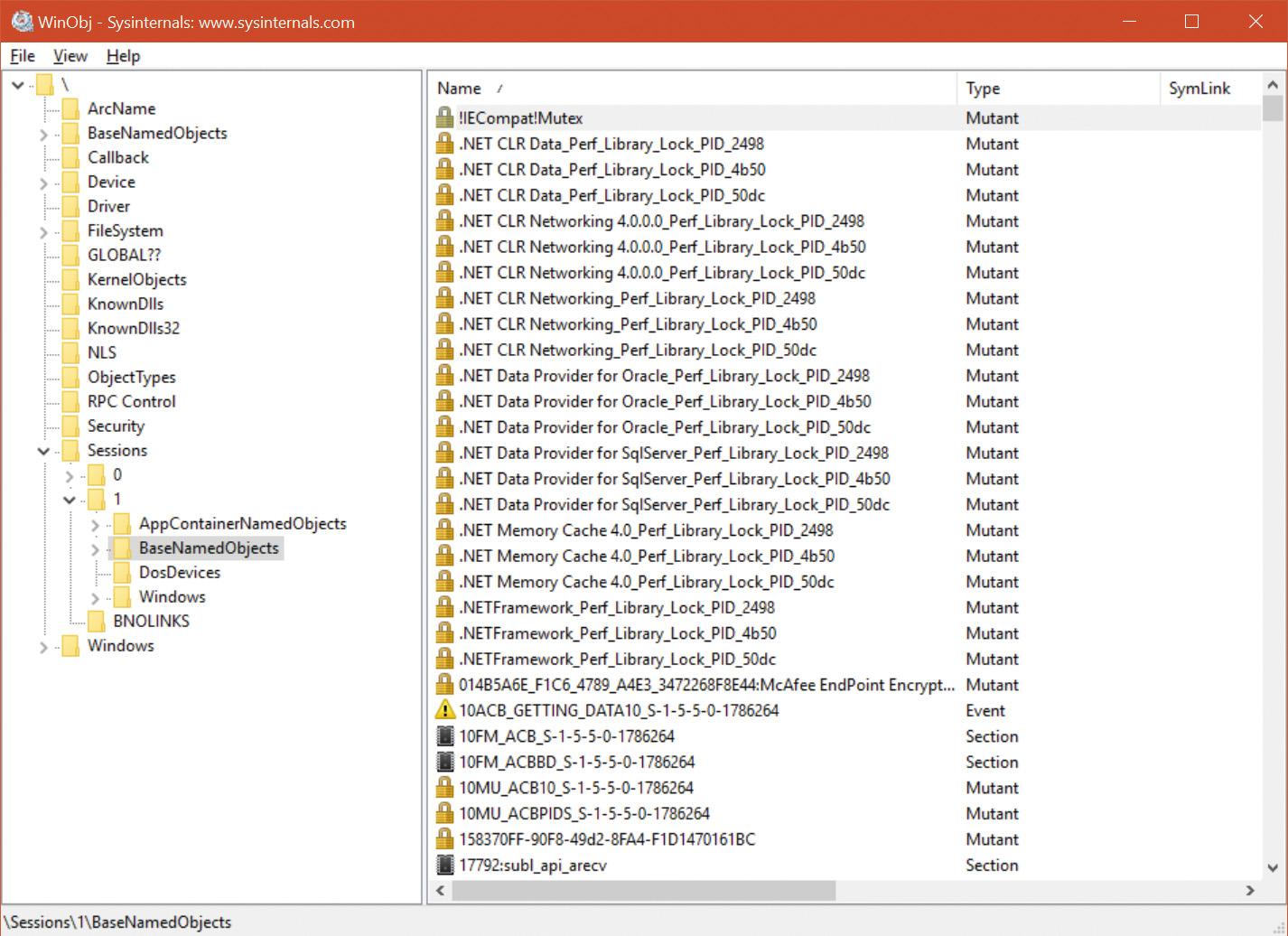
The output shows the package full name, executable directory, AppContainer SID, publisher information, version, and list of capabilities. Also shown is the LowBox number, which is just a local index of the app.

Lastly, you can inspect these properties in the kernel debugger with the !token command.

Some UWP apps are called *trusted*, and although they use the Windows Runtime platform like other UWP apps, they do not run inside an AppContainer, and have an integrity level higher than Low. The canonical example is the System Settings app (%SystemRoot%\ImmersiveControlPanel\SystemSettings.exe); this seems reasonable, as the Settings app must be able to make changes to the system that would be impossible to do from an AppContainer-hosted process. If you look at its token, you will see the same three attributes—PKG, SYSAPPID, and PKGHOSTID—which confirm that it’s still a packaged application, even without the AppContainer token present.

##### **AppContainer and object namespace**

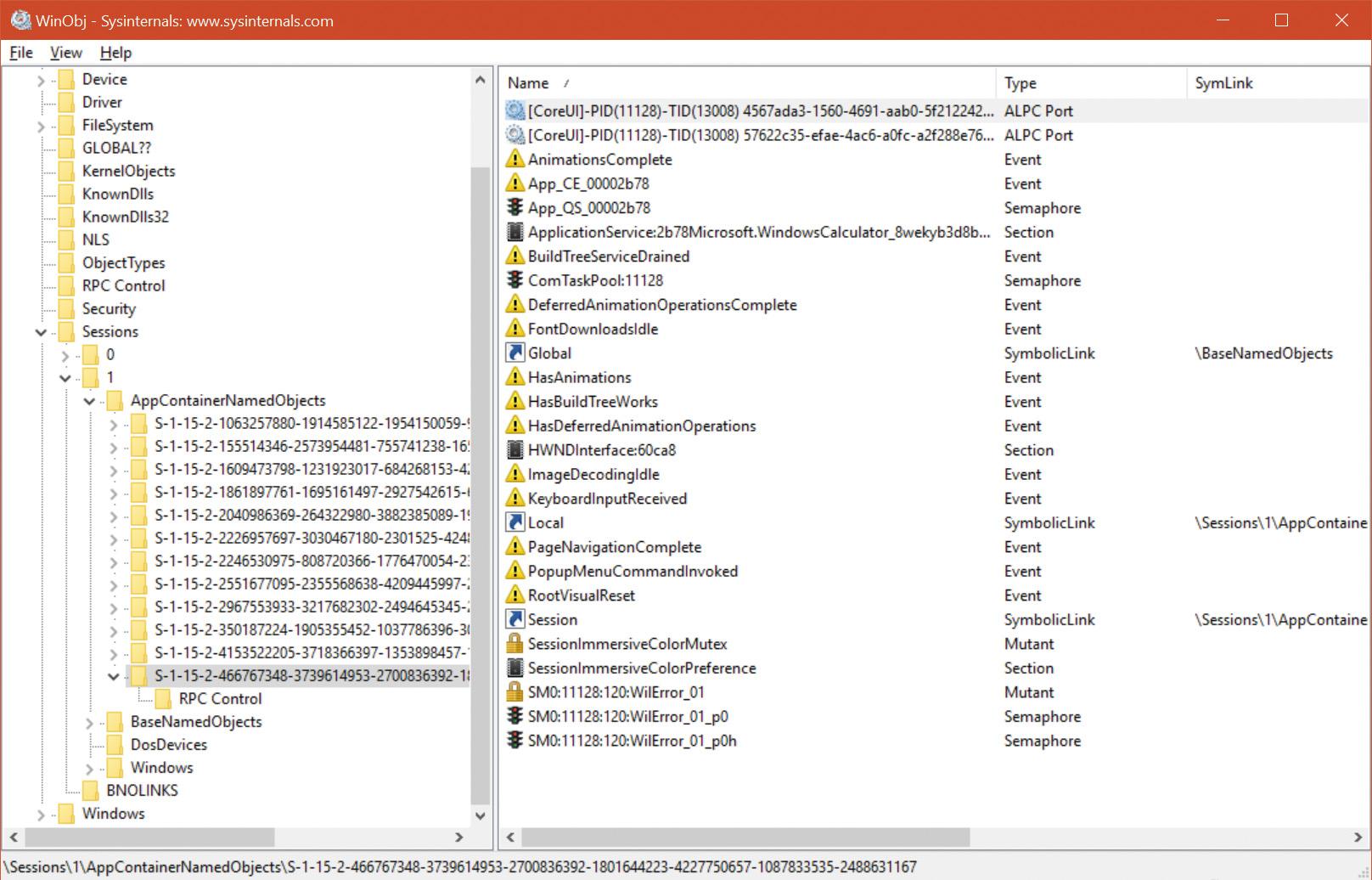
Desktop applications can easily share kernel objects by name. For example, suppose process A creates an event object by calling CreateEvent(Ex) with the name MyEvent. It gets back a handle it can later use to manipulate the event. Process B running in the same session can call CreateEvent(Ex) or OpenEvent with the same name, MyEvent, and (assuming it has appropriate permissions, which is usually the case if running under the same session) get back another handle to the same underlying event object. Now if process A calls SetEvent on the event object while process B was blocked in a call to WaitForSingleObject on its event handle, process B’s waiting thread would be released because it’s the same event object. This sharing works because named objects are created in the object manager directory \Sessions\x\BaseNamedObjects, as shown in [Figure 7-18](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig18) with the WinObj Sysinternals tool.



**FIGURE 7-18** Object manager directory for named objects.

Furthermore, desktop apps can share objects between sessions by using a name prefixed with Global\. This creates the object in the session 0 object directory located in \BaseNamedObjects (refer to [Figure 7-18](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig18)).

AppContainer-based processes have their root object namespace under \Sessions\x\AppContainerNamedObjects\<AppContainerSID>. Since every AppContainer has a different AppContainer SID, there is no way two UWP apps can share kernel objects. The ability to create a named kernel object in the session 0 object namespace is not allowed for AppContainer processes. [Figure 7-19](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig19) shows the object manager’s directory for the Windows UWP Calculator app.



**FIGURE 7-19** Object manager directory for Calculator.

UWP apps that want to share data can do so using well-defined contracts, managed by the Windows Runtime. (See the MSDN documentation for more information.)

Sharing kernel objects between desktop apps and UWP apps is possible, and often done by broker services. For example, when requesting access to a file in the Documents folder (and getting the right capability validated) from the file picker broker, the UWP app will receive a file handle that it can use for reads and writes directly, without the cost of marshalling requests back and forth. This is achieved by having the broker duplicate the file handle it obtained directly in the handle table of the UWP application. (More information on handle duplication appears in Chapter 8 in Part 2.) To simplify things even further, the ALPC subsystem (also described in Chapter 8) allows the automatic transfer of handles in this way through ALPC handle attributes. and the Remote Procedure Call (RPC) services that use ALPC as their underlying protocol can use this functionality as part of their interfaces. Marshallable handles in the IDL file will automatically be transferred in this way through the ALPC subsystem.

Outside of official broker RPC services, a desktop app can create a named (or even unnamed) object normally, and then use the DuplicateHandle function to inject a handle to the same object into the UWP process manually. This works because desktop apps typically run with medium integrity level and there’s nothing preventing them from duplicating handles into UWP processes—only the other way around.

Image Note

Communication between a desktop app and a UWP is not usually required because a store app cannot have a desktop app companion, and cannot rely on such an app to exist on the device. The capability to inject handles into a UWP app may be needed in specialized cases such as using the desktop bridge (Centennial) to convert a desktop app to a UWP app and communicate with another desktop app that is known to exist.

##### **AppContainer handles**

In a typical Win32 application, the presence of the session-local and global BaseNamedObjects directory is guaranteed by the Windows subsystem, as it creates this on boot and session creation. Unfortunately, the AppContainerBaseNamedObjects directory is actually created by the launch application itself. In the case of UWP activation, this is the trusted DComLaunch service, but recall that not all AppContainers are necessarily tied to UWP. They can also be manually created through the right process-creation attributes. (See [Chapter 3](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#ch03) for more information on which ones to use.) In this case, it’s possible for an untrusted application to have created the object directory (and required symbolic links within it), which would result in the ability for this application to close the handles from underneath the AppContainer application. Even without malicious intent, the original launching application might exit, cleaning up its handles and destroying the AppContainer-specific object directory. To avoid this situation, AppContainer tokens have the ability to store an array of handles that are guaranteed to exist throughout the lifetime of any application using the token. These handles are initially passed in when the AppContainer token is being created (through NtCreateLowBoxToken) and are duplicated as kernel handles.

Similar to the per-AppContainer atom table, a special SEP\_CACHED\_HANDLES\_ENTRY structure is used, this time based on a hash table that’s stored in the logon session structure for this user. (See the “[Logon](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07lev1sec10)” section later in this chapter for more information on logon sessions.) This structure contains an array of kernel handles that have been duplicated during the creation of the AppContainer token. They will be closed either when this token is destroyed (because the application is exiting) or when the user logs off (which will result in tearing down the logon session).

EXPERIMENT: Viewing token stored handles

To view token stored handles, follow these steps:

**1.** Run Calculator and launch local kernel debugging.

**2.** Search for the calculator process:

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p706pro01a)

lkd> !process 0 1 calculator.exe

PROCESS ffff828cc9ed1080

SessionId: 1 Cid: 4bd8 Peb: d040bbc000 ParentCid: 03a4

DeepFreeze

DirBase: 5fccaa000 ObjectTable: ffff950ad9fa2800 HandleCount:

<Data Not Accessible>

Image: Calculator.exe

VadRoot ffff828cd2c9b6a0 Vads 168 Clone 0 Private 2938. Modified 3332.

Locked 0.

DeviceMap ffff950aad2cd2f0

Token **ffff950adb313060**

ElapsedTime 1 Day 08:01:47.018

UserTime 00:00:00.015

KernelTime 00:00:00.031

QuotaPoolUsage[PagedPool] 465880

QuotaPoolUsage[NonPagedPool] 23288

Working Set Sizes (now,min,max) (7434, 50, 345) (29736KB, 200KB, 1380KB)

PeakWorkingSetSize 11097

VirtualSize 303 Mb

PeakVirtualSize 314 Mb

PageFaultCount 21281

MemoryPriority BACKGROUND

BasePriority 8

CommitCharge 4925

Job ffff828cd4914060

**3.** Dump the token using the dt command. (Remember to mask the lower 3 or 4 bits if they are not zero.)

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p707pro01a)

lkd> dt nt!\_token ffff950adb313060

+0x000 TokenSource : \_TOKEN\_SOURCE

+0x010 TokenId : \_LUID

+0x018 AuthenticationId : \_LUID

+0x020 ParentTokenId : \_LUID

...

+0x0c8 TokenFlags : 0x4a00

+0x0cc TokenInUse : 0x1 ''

+0x0d0 IntegrityLevelIndex : 1

+0x0d4 MandatoryPolicy : 1

+0x0d8 LogonSession : 0xffff950a'b4bb35c0 \_SEP\_LOGON\_SESSION\_REFERENCES

+0x0e0 OriginatingLogonSession : \_LUID

+0x0e8 SidHash : \_SID\_AND\_ATTRIBUTES\_HASH

+0x1f8 RestrictedSidHash : \_SID\_AND\_ATTRIBUTES\_HASH

+0x308 pSecurityAttributes : 0xffff950a'e4ff57f0 \_AUTHZBASEP\_SECURITY\_

ATTRIBUTES\_INFORMATION

+0x310 Package : 0xffff950a'e00ed6d0 Void

+0x318 Capabilities : 0xffff950a'e8e8fbc0 \_SID\_AND\_ATTRIBUTES

+0x320 CapabilityCount : 1

+0x328 CapabilitiesHash : \_SID\_AND\_ATTRIBUTES\_HASH

+0x438 LowboxNumberEntry : 0xffff950a'b3fd55d0 \_SEP\_LOWBOX\_NUMBER\_ENTRY

**+0x440 LowboxHandlesEntry : 0xffff950a'e6ff91d0 \_SEP\_LOWBOX\_HANDLES\_ENTRY**

+0x448 pClaimAttributes : (null)

...

**4.** Dump the LowboxHandlesEntry member:

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p707pro02a)

lkd> dt nt!\_sep\_lowbox\_handles\_entry 0xffff950a'e6ff91d0

+0x000 HashEntry : \_RTL\_DYNAMIC\_HASH\_TABLE\_ENTRY

+0x018 ReferenceCount : 0n10

+0x020 PackageSid : 0xffff950a'e6ff9208 Void

+0x028 HandleCount : 6

**+0x030 Handles : 0xffff950a'e91d8490 -> 0xffffffff'800023cc Void**

**5.** There are six handles. Let’s dump their values:

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p707pro03a)

lkd> dq 0xffff950ae91d8490 L6

ffff950a'e91d8490 ffffffff'800023cc ffffffff'80001e80

ffff950a'e91d84a0 ffffffff'80004214 ffffffff'8000425c

ffff950a'e91d84b0 ffffffff'800028c8 ffffffff'80001834

**6.** You can see that these handles are kernel handles—that is, handle values starting with 0xffffffff (64 bit). Now you can use the !handle command to look at individual handles. Here are two examples from the six handles above:

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p707pro04a)

lkd> !handle ffffffff'80001e80

PROCESS ffff828cd71b3600

SessionId: 1 Cid: 27c4 Peb: 3fdfb2f000 ParentCid: 2324

DirBase: 80bb85000 ObjectTable: ffff950addabf7c0 HandleCount:

<Data Not Accessible>

Image: windbg.exe

Kernel handle Error reading handle count.

80001e80: Object: ffff950ada206ea0 GrantedAccess: 0000000f (Protected)

(Inherit) (Audit) Entry: ffff950ab5406a00

Object: ffff950ada206ea0 Type: (ffff828cb66b33b0) Directory

ObjectHeader: ffff950ada206e70 (new version)

HandleCount: 1 PointerCount: 32770

Directory Object: ffff950ad9a62950 Name: RPC Control

Hash Address Type Name

---- ------- ---- ----

23 ffff828cb6ce6950 ALPC Port

OLE376512B99BCCA5DE4208534E7732

lkd> !handle ffffffff'800028c8

PROCESS ffff828cd71b3600

SessionId: 1 Cid: 27c4 Peb: 3fdfb2f000 ParentCid: 2324

DirBase: 80bb85000 ObjectTable: ffff950addabf7c0 HandleCount: <Data

Not Accessible>

Image: windbg.exe

Kernel handle Error reading handle count.

800028c8: Object: ffff950ae7a8fa70 GrantedAccess: 000f0001 (Audit) Entry:

ffff950acc426320

Object: ffff950ae7a8fa70 Type: (ffff828cb66296f0) SymbolicLink

ObjectHeader: ffff950ae7a8fa40 (new version)

HandleCount: 1 PointerCount: 32769

Directory Object: ffff950ad9a62950 Name: Session

Flags: 00000000 ( Local )

Target String is '\Sessions\1\AppContainerNamedObjects

\S-1-15-2-466767348-3739614953-2700836392-1801644223-4227750657

-1087833535-2488631167'

Finally, because the ability to restrict named objects to a particular object directory namespace is a valuable security tool for sandboxing named object access, the upcoming (at the time of this writing) Windows 10 Creators Update includes an additional token capability called *BNO isolation* (where *BNO* refers to BaseNamedObjects). Using the same SEP\_CACHE\_HANDLES\_ENTRY structure, a new field, BnoIsolationHandlesEntry, is added to the TOKEN structure, with the type set to SepCachedHandlesEntryBnoIsolation instead of SepCachedHandlesEntryLowbox. To use this feature, a special process attribute must be used (see [Chapter 3](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#ch03) for more information), which contains an isolation prefix and a list of handles. At this point, the same LowBox mechanism is used, but instead of an AppContainer SID object directory, a directory with the prefix indicated in the attribute is used.

##### **Brokers**

Because AppContainer processes have almost no permissions except for those implicitly granted with capabilities, some common operations cannot be performed directly by the AppContainer and require help. (There are no capabilities for these, as these are too low level to be visible to users in the store, and difficult to manage.) Some examples include selecting files using the common File Open dialog box or printing with a Print dialog box. For these and other similar operations, Windows provides helper processes, called *brokers*, managed by the system broker process, RuntimeBroker.exe.

An AppContainer process that requires any of these services communicates with the Runtime Broker through a secure ALPC channel and Runtime Broker initiates the creation of the requested broker process. Examples are %SystemRoot%\PrintDialog\PrintDialog.exe and %SystemRoot%\System32\PickerHost.exe.

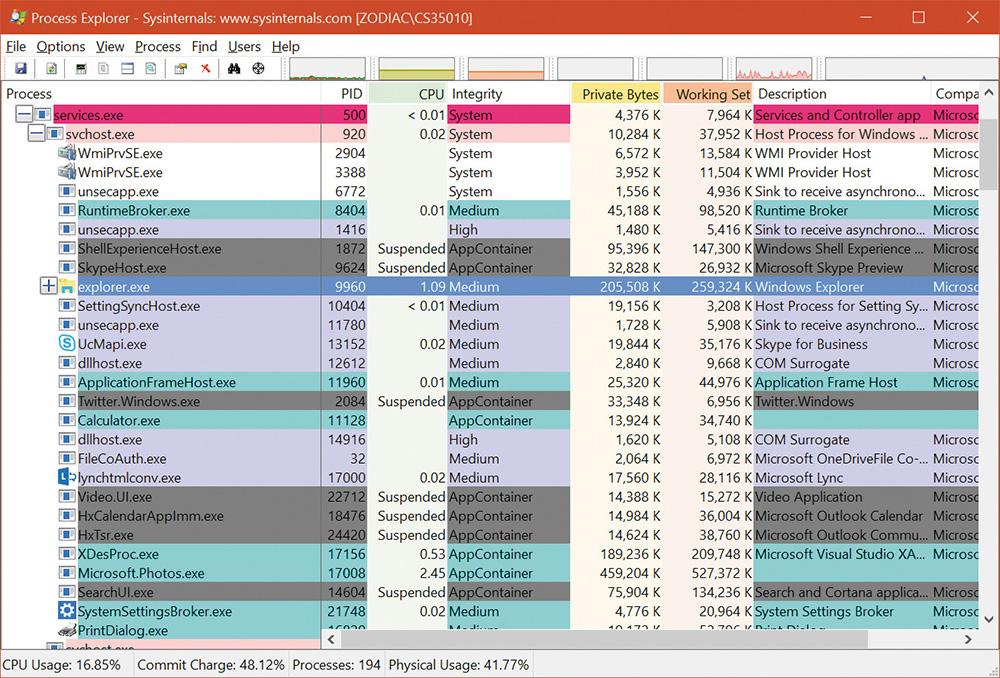
EXPERIMENT: Brokers

The following steps show how broker processes are launched and terminated:

**1.** Click the **Start** button, type **Photos**, and select the **Photos** option to run the built-in Windows 10 Photos application.

**2.** Open Process Explorer, switch the process list to a tree view, and locate the Microsoft.Photos.exe process. Place both windows side by side.

**3.** In the Photos app, select a picture file, and click **Print** in the top ellipsis menu or right-click the picture and choose **Print** from the menu that appears. The Print dialog box should open, and Process Explorer should show the newly created broker (PrintDialog.exe). Notice they are all children of the same Svchost process. (All UWP processes are launched by the DCOMLaunch service hosted inside that process.)



**4.** Close the Print dialog box. The PrintDialog.exe process should exit.

### **LOGON**

Interactive logon (as opposed to network logon) occurs through the interaction of the following:

Image The logon process (Winlogon.exe)

Image The logon user interface process (LogonUI.exe) and its credential providers

Image Lsass.exe

Image One or more authentication packages

Image SAM or Active Directory

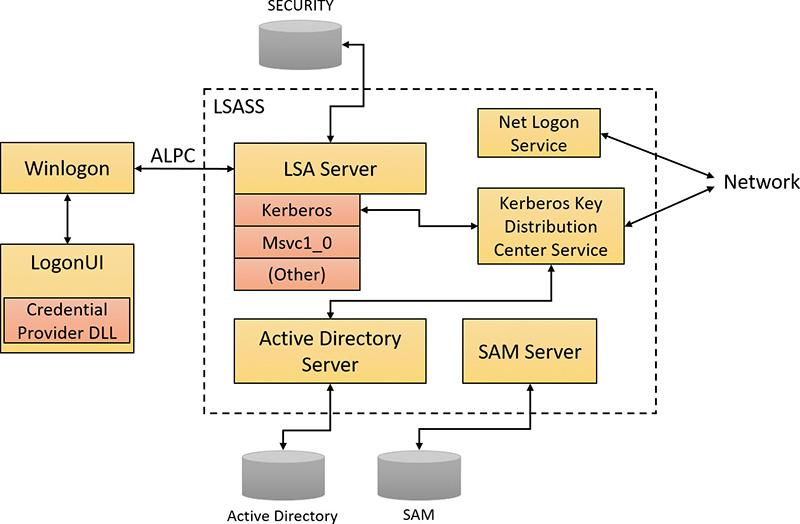
Authentication packages are DLLs that perform authentication checks. Kerberos is the Windows authentication package for interactive logon to a domain. MSV1\_0 is the Windows authentication package for interactive logon to a local computer, for domain logons to trusted pre–Windows 2000 domains, and for times when no domain controller is accessible.

Winlogon is a trusted process responsible for managing security-related user interactions. It coordinates logon, starts the user’s first process at logon, and handles logoff. It also manages various other operations relevant to security, including launching LogonUI for entering passwords at logon, changing passwords, and locking and unlocking the workstation. The Winlogon process must ensure that operations relevant to security aren’t visible to any other active processes. For example, Winlogon guarantees that an untrusted process can’t get control of the desktop during one of these operations and thus gain access to the password.

Winlogon relies on the credential providers installed on the system to obtain a user’s account name or password. Credential providers are COM objects located inside DLLs. The default providers are authui.dll, SmartcardCredentialProvider.dll, and FaceCredentialProvider.dll, which support password, smartcard PIN, and face-recognition authentication, respectively. Allowing other credential providers to be installed enables Windows to use different user-identification mechanisms. For example, a third party might supply a credential provider that uses a thumbprint-recognition device to identify users and extract their passwords from an encrypted database. Credential providers are listed in HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Authentication\Credential Providers, where each subkey identifies a credential provider class by its COM CLSID. (The CLSID itself must be registered at HKCR\CLSID like any other COM class.) You can use the CPlist.exe tool provided with the downloadable resources for this book to list the credential providers with their CLSID, friendly name, and implementation DLL.

To protect Winlogon’s address space from bugs in credential providers that might cause the Winlogon process to crash (which, in turn, will result in a system crash, because Winlogon is considered a critical system process), a separate process, LogonUI.exe, is used to actually load the credential providers and display the Windows logon interface to users. This process is started on demand whenever Winlogon needs to present a user interface to the user, and it exits after the action has finished. It also allows Winlogon to simply restart a new LogonUI process should it crash for any reason.

Winlogon is the only process that intercepts logon requests from the keyboard. These are sent through an RPC message from Win32k.sys. Winlogon immediately launches the LogonUI application to display the user interface for logon. After obtaining a user name and password from credential providers, Winlogon calls Lsass to authenticate the user attempting to log on. If the user is authenticated, the logon process activates a logon shell on behalf of that user. The interaction between the components involved in logon is illustrated in [Figure 7-20](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig20).



**FIGURE 7-20** Components involved in logon.

In addition to supporting alternative credential providers, LogonUI can load additional network provider DLLs that need to perform secondary authentication. This capability allows multiple network providers to gather identification and authentication information all at one time during normal logon. A user logging on to a Windows system might simultaneously be authenticated on a Linux server. That user would then be able to access resources of the UNIX server from the Windows machine without requiring additional authentication. Such a capability is known as one form of single sign-on.

#### **Winlogon initialization**

During system initialization, before any user applications are active, Winlogon performs the following steps to ensure that it controls the workstation once the system is ready for user interaction:

**1.** It creates and opens an interactive window station (for example, \Sessions\1\Windows\WindowStations\WinSta0 in the object manager namespace) to represent the keyboard, mouse, and monitor. Winlogon creates a security descriptor for the station that has one and only one ACE containing only the system SID. This unique security descriptor ensures that no other process can access the workstation unless explicitly allowed by Winlogon.

**2.** It creates and opens two desktops: an application desktop (\Sessions\1\Windows\WinSta0\Default, also known as the interactive desktop) and a Winlogon desktop (\Sessions\1\Windows\WinSta0\Winlogon, also known as the Secure Desktop). The security on the Winlogon desktop is created so that only Winlogon can access that desktop. The other desktop allows both Winlogon and users to access it. This arrangement means that any time the Winlogon desktop is active, no other process has access to any active code or data associated with the desktop. Windows uses this feature to protect the secure operations that involve passwords and locking and unlocking the desktop.

**3.** Before anyone logs on to a computer, the visible desktop is Winlogon’s. After a user logs on, pressing the SAS sequence (by default, Ctrl+Alt+Del) switches the desktop from Default to Winlogon and launches LogonUI. (This explains why all the windows on your interactive desktop seem to disappear when you press Ctrl+Alt+Del, and then return when you dismiss the Windows Security dialog box.) Thus, the SAS always brings up a Secure Desktop controlled by Winlogon.

**4.** It establishes an ALPC connection with Lsass. This connection will be used for exchanging information during logon, logoff, and password operations, and is made by calling LsaRegisterLogonProcess.

**5.** It registers the Winlogon RPC message server, which listens for SAS, logoff, and workstation lock notifications from Win32k. This measure prevents Trojan horse programs from gaining control of the screen when the SAS is entered.

Image Note

The Wininit process performs steps similar to steps 1 and 2 to allow legacy interactive services running on session 0 to display windows, but it does not perform any other steps because session 0 is not available for user logon.

How SAS is implemented

The SAS is secure because no application can intercept the Ctrl+Alt+Del keystroke combination or prevent Winlogon from receiving it. Win32k.sys reserves the Ctrl+Alt+Del key combination so that whenever the Windows input system (implemented in the raw input thread in Win32k) sees the combination, it sends an RPC message to Winlogon’s message server, which listens for such notifications. The keystrokes that map to a registered hot key are not sent to any process other than the one that registered it, and only the thread that registered a hot key can unregister it, so a Trojan horse application cannot deregister Winlogon’s ownership of the SAS.

A Windows function, SetWindowsHookEx, enables an application to install a hook procedure that’s invoked every time a keystroke is pressed, even before hot keys are processed, and allows the hook to squash keystrokes. However, the Windows hot key processing code contains a special case for Ctrl+Alt+Del that disables hooks so that the keystroke sequence can’t be intercepted. In addition, if the interactive desktop is locked, only hot keys owned by Winlogon are processed.

Once the Winlogon desktop is created during initialization, it becomes the active desktop. When the Winlogon desktop is active, it is always locked. Winlogon unlocks its desktop only to switch to the application desktop or the screen-saver desktop. (Only the Winlogon process can lock or unlock a desktop.)

#### **User logon steps**

Logon begins when a user presses the SAS (Ctrl+Alt+Del). After the SAS is pressed, Winlogon starts LogonUI, which calls the credential providers to obtain a user name and password. Winlogon also creates a unique local logon SID for this user, which it assigns to this instance of the desktop (keyboard, screen, and mouse). Winlogon passes this SID to Lsass as part of the LsaLogonUser call. If the user is successfully logged on, this SID will be included in the logon process token—a step that protects access to the desktop. For example, another logon to the same account but on a different system will be unable to write to the first machine’s desktop because this second logon won’t be in the first logon’s desktop token.

When the user name and password have been entered, Winlogon retrieves a handle to a package by calling the Lsass function LsaLookupAuthenticationPackage. Authentication packages are listed in the registry under HKLM\SYSTEM\CurrentControlSet\Control\Lsa. Winlogon passes logon information to the authentication package via LsaLogonUser. Once a package authenticates a user, Winlogon continues the logon process for that user. If none of the authentication packages indicates a successful logon, the logon process is aborted.

Windows uses two standard authentication packages for interactive username/password-based logons:

Image **MSV1\_0** The default authentication package on a stand-alone Windows system is MSV1\_0 (Msv1\_0.dll), an authentication package that implements LAN Manager 2 protocol. Lsass also uses MSV1\_0 on domain-member computers to authenticate to pre–Windows 2000 domains and computers that can’t locate a domain controller for authentication. (Computers that are disconnected from the network fall into this latter category.)

Image **Kerberos** The Kerberos authentication package, Kerberos.dll, is used on computers that are members of Windows domains. The Windows Kerberos package, with the cooperation of Kerberos services running on a domain controller, supports the Kerberos protocol. This protocol is based on Internet RFC 1510. (Visit the Internet Engineering Task Force [IETF] website at [*http://www.ietf.org*](http://www.ietf.org/) for detailed information on the Kerberos standard.)

##### **MSV1\_0**

The MSV1\_0 authentication package takes the user name and a hashed version of the password and sends a request to the local SAM to retrieve the account information, which includes the hashed password, the groups to which the user belongs, and any account restrictions. MSV1\_0 first checks the account restrictions, such as hours or type of accesses allowed. If the user can’t log on because of the restrictions in the SAM database, the logon call fails and MSV1\_0 returns a failure status to the LSA.

MSV1\_0 then compares the hashed password and user name to that obtained from the SAM. In the case of a cached domain logon, MSV1\_0 accesses the cached information by using Lsass functions that store and retrieve “secrets” from the LSA database (the SECURITY hive of the registry). If the information matches, MSV1\_0 generates a LUID for the logon session and creates the logon session by calling Lsass, associating this unique identifier with the session and passing the information needed to ultimately create an access token for the user. (Recall that an access token includes the user’s SID, group SIDs, and assigned privileges.)

Image Note

MSV1\_0 does not cache a user’s entire password hash in the registry because that would enable someone with physical access to the system to easily compromise a user’s domain account and gain access to encrypted files and to network resources the user is authorized to access. Instead, it caches half of the hash. The cached half-hash is sufficient to verify that a user’s password is correct, but it isn’t sufficient to gain access to EFS keys and to authenticate as the user on a domain because these actions require the full hash.

If MSV1\_0 needs to authenticate using a remote system, as when a user logs on to a trusted pre–Windows 2000 domain, MSV1\_0 uses the Netlogon service to communicate with an instance of Netlogon on the remote system. Netlogon on the remote system interacts with the MSV1\_0 authentication package on that system, passing back authentication results to the system on which the logon is being performed.

##### **Kerberos**

The basic control flow for Kerberos authentication is the same as the flow for MSV1\_0. However, in most cases, domain logons are performed from member workstations or servers rather than on a domain controller, so the authentication package must communicate across the network as part of the authentication process. The package does so by communicating via the Kerberos TCP/IP port (port 88) with the Kerberos service on a domain controller. The Kerberos Key Distribution Center service (Kdcsvc.dll), which implements the Kerberos authentication protocol, runs in the Lsass process on domain controllers.

After validating hashed user-name and password information with Active Directory’s user account objects (using the Active Directory server Ntdsa.dll), Kdcsvc returns domain credentials to Lsass, which returns the result of the authentication and the user’s domain logon credentials (if the logon was successful) across the network to the system where the logon is taking place.

Image Note

This description of Kerberos authentication is highly simplified, but it highlights the roles of the various components involved. Although the Kerberos authentication protocol plays a key role in distributed domain security in Windows, its details are outside the scope of this book.

After a logon has been authenticated, Lsass looks in the local policy database for the user’s allowed access, including interactive, network, batch, or service process. If the requested logon doesn’t match the allowed access, the logon attempt will be terminated. Lsass deletes the newly created logon session by cleaning up any of its data structures and then returns failure to Winlogon, which in turn displays an appropriate message to the user. If the requested access is allowed, Lsass adds the appropriate additional security IDs (such as Everyone, Interactive, and the like). It then checks its policy database for any granted privileges for all the SIDs for this user and adds these privileges to the user’s access token.

When Lsass has accumulated all the necessary information, it calls the executive to create the access token. The executive creates a primary access token for an interactive or service logon and an impersonation token for a network logon. After the access token is successfully created, Lsass duplicates the token, creating a handle that can be passed to Winlogon, and closes its own handle. If necessary, the logon operation is audited. At this point, Lsass returns success to Winlogon along with a handle to the access token, the LUID for the logon session, and the profile information, if any, that the authentication package returned.

EXPERIMENT: Listing active logon sessions

As long as at least one token exists with a given logon session LUID, Windows considers the logon session to be active. You can use the LogonSessions tool from Sysinternals, which uses the LsaEnumerateLogonSessions function (documented in the Windows SDK) to list the active logon sessions:

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p715pro01a)

C:\WINDOWS\system32>logonsessions

LogonSessions v1.4 - Lists logon session information

Copyright (C) 2004-2016 Mark Russinovich

Sysinternals - www.sysinternals.com

[0] Logon session 00000000:000003e7:

User name: WORKGROUP\ZODIAC$

Auth package: NTLM

Logon type: (none)

Session: 0

Sid: S-1-5-18

Logon time: 09-Dec-16 15:22:31

Logon server:

DNS Domain:

UPN:

[1] Logon session 00000000:0000cdce:

User name:

Auth package: NTLM

Logon type: (none)

Session: 0

Sid: (none)

Logon time: 09-Dec-16 15:22:31

Logon server:

DNS Domain:

UPN:

[2] Logon session 00000000:000003e4:

User name: WORKGROUP\ZODIAC$

Auth package: Negotiate

Logon type: Service

Session: 0

Sid: S-1-5-20

Logon time: 09-Dec-16 15:22:31

Logon server:

DNS Domain:

UPN:

[3] Logon session 00000000:00016239:

User name: Window Manager\DWM-1

Auth package: Negotiate

Logon type: Interactive

Session: 1

Sid: S-1-5-90-0-1

Logon time: 09-Dec-16 15:22:32

Logon server:

DNS Domain:

UPN:

[4] Logon session 00000000:00016265:

User name: Window Manager\DWM-1

Auth package: Negotiate

Logon type: Interactive

Session: 1

Sid: S-1-5-90-0-1

Logon time: 09-Dec-16 15:22:32

Logon server:

DNS Domain:

UPN:

[5] Logon session 00000000:000003e5:

User name: NT AUTHORITY\LOCAL SERVICE

Auth package: Negotiate

Logon type: Service

Session: 0

Sid: S-1-5-19

Logon time: 09-Dec-16 15:22:32

Logon server:

DNS Domain:

UPN:

...

[8] Logon session 00000000:0005c203:

User name: NT VIRTUAL MACHINE\AC9081B6-1E96-4BC8-8B3B-C609D4F85F7D

Auth package: Negotiate

Logon type: Service

Session: 0

Sid: S-1-5-83-1-2895151542-1271406230-163986315-2103441620

Logon time: 09-Dec-16 15:22:35

Logon server:

DNS Domain:

UPN:

[9] Logon session 00000000:0005d524:

User name: NT VIRTUAL MACHINE\B37F4A3A-21EF-422D-8B37-AB6B0A016ED8

Auth package: Negotiate

Logon type: Service

Session: 0

Sid: S-1-5-83-1-3011463738-1110254063-1806382987-3631087882

Logon time: 09-Dec-16 15:22:35

Logon server:

DNS Domain:

UPN:

...

[12] Logon session 00000000:0429ab2c:

User name: IIS APPPOOL\DefaultAppPool

Auth package: Negotiate

Logon type: Service

Session: 0

Sid: S-1-5-82-3006700770-424185619-1745488364-794895919-4004696415

Logon time: 09-Dec-16 22:33:03

Logon server:

DNS Domain:

UPN:

Information reported for a session includes the SID and name of the user associated with the session, as well as the session’s authentication package and logon time. Note that the Negotiate authentication package, seen in logon sessions 2 and 9 in the preceding output, will attempt to authenticate via Kerberos or NTLM, depending on which is most appropriate for the authentication request.

The LUID for a session is displayed on the Logon Session line of each session block. Using the Handle.exe utility (also from Sysinternals), you can find the tokens that represent a particular logon session. For example, to find the tokens for logon session 8 in the output just shown, you could enter this command:

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p717pro01a)

C:\WINDOWS\system32>handle -a 5c203

Nthandle v4.1 - Handle viewer

Copyright (C) 1997-2016 Mark Russinovich

Sysinternals - www.sysinternals.com

System pid: 4 type: Directory 1274: \Sessions\0\

DosDevices\00000000-0005c203

lsass.exe pid: 496 type: Token D7C: NT VIRTUAL MACHINE\

AC9081B6-1E96-4BC8-8B3B-C609D4F85F7D:5c203

lsass.exe pid: 496 type: Token 2350: NT VIRTUAL MACHINE\

AC9081B6-1E96-4BC8-8B3B-C609D4F85F7D:5c203

lsass.exe pid: 496 type: Token 2390: NT VIRTUAL MACHINE\

AC9081B6-1E96-4BC8-8B3B-C609D4F85F7D:5c203

svchost.exe pid: 900 type: Token 804: NT VIRTUAL MACHINE\

AC9081B6-1E96-4BC8-8B3B-C609D4F85F7D:5c203

svchost.exe pid: 1468 type: Token 10EC: NT VIRTUAL MACHINE\

AC9081B6-1E96-4BC8-8B3B-C609D4F85F7D:5c203

vmms.exe pid: 4380 type: Token A34: NT VIRTUAL MACHINE\

AC9081B6-1E96-4BC8-8B3B-C609D4F85F7D:5c203

vmcompute.exe pid: 6592 type: Token 200: NT VIRTUAL MACHINE\

AC9081B6-1E96-4BC8-8B3B-C609D4F85F7D:5c203

vmwp.exe pid: 7136 type: WindowStation 168: \Windows\WindowStations\

Service-0x0-5c203$

vmwp.exe pid: 7136 type: WindowStation 170: \Windows\WindowStations\

Service-0x0-5c203$

Winlogon then looks in the registry at the value HKLM\SOFTWARE\Microsoft\Windows NT\Current Version\Winlogon\Userinit and creates a process to run whatever the value of that string is. (This value can be several EXEs separated by commas.) The default value is Userinit.exe, which loads the user profile and then creates a process to run whatever the value of HKCU\SOFTWARE\Microsoft\Windows NT\Current Version\Winlogon\Shell is, if that value exists. That value does not exist by default, however. If it doesn’t exist, Userinit.exe does the same for HKLM\SOFTWARE\Microsoft\Windows NT\Current Version\Winlogon\Shell, which defaults to Explorer.exe. Userinit then exits (which is why Explorer.exe shows up as having no parent when examined in Process Explorer). For more information on the steps followed during the user logon process, see Chapter 11 in Part 2.

#### **Assured authentication**

A fundamental problem with password-based authentication is that passwords can be revealed or stolen and used by malicious third parties. Windows includes a mechanism that tracks the authentication strength of how a user authenticated with the system, which allows objects to be protected from access if a user did not authenticate securely. (Smartcard authentication is considered to be a stronger form of authentication than password authentication.)

On systems that are joined to a domain, the domain administrator can specify a mapping between an object identifier (OID) (a unique numeric string representing a specific object type) on a certificate used for authenticating a user (such as on a smartcard or hardware security token) and a SID that is placed into the user’s access token when the user successfully authenticates with the system. An ACE in a DACL on an object can specify such a SID be part of a user’s token in order for the user to gain access to the object. Technically, this is known as a *group claim*. In other words, the user is claiming membership in a particular group, which is allowed certain access rights on specific objects, with the claim based upon the authentication mechanism. This feature is not enabled by default, and it must be configured by the domain administrator in a domain with certificate-based authentication.

Assured authentication builds on existing Windows security features in a way that provides a great deal of flexibility to IT administrators and anyone concerned with enterprise IT security. The enterprise decides which OIDs to embed in the certificates it uses for authenticating users and the mapping of particular OIDs to Active Directory universal groups (SIDs). A user’s group membership can be used to identify whether a certificate was used during the logon operation. Different certificates can have different issuance policies and, thus, different levels of security, which can be used to protect highly sensitive objects (such as files or anything else that might have a security descriptor).

Authentication protocols (APs) retrieve OIDs from certificates during certificate-based authentication. These OIDs must be mapped to SIDs, which are in turn processed during group membership expansion, and placed in the access token. The mapping of OID to universal group is specified in Active Directory.

As an example, an organization might have several certificate-issuance policies named Contractor, Full Time Employee, and Senior Management, which map to the universal groups Contractor-Users, FTE-Users, and SM-Users, respectively. A user named Abby has a smartcard with a certificate issued using the Senior Management issuance policy. When she logs in using her smartcard, she receives an additional group membership (which is represented by a SID in her access token) indicating that she is a member of the SM-Users group. Permissions can be set on objects (using an ACL) such that only members of the FTE-Users or SM-Users group (identified by their SIDs within an ACE) are granted access. If Abby logs in using her smartcard, she can access those objects, but if she logs in with just her user name and password (without the smartcard), she cannot access those objects because she will not have either the FTE-Users or SM-Users group in her access token. A user named Toby who logs in with a smartcard that has a certificate issued using the Contractor issuance policy would not be able to access an object that has an ACE requiring FTE-Users or SM-Users group membership.

#### **Windows Biometric Framework**

Windows provides a standardized mechanism for supporting certain types of biometric devices, such as fingerprint scanners, used to enable user identification via a fingerprint swipe: the Windows Biometric Framework (WBF). Like many other such frameworks, the WBF was developed to isolate the various functions involved in supporting such devices, so as to minimize the code required to implement a new device.

The primary components of the WBF are shown in [Figure 7-21](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig21). Except as noted in the following list, all of these components are supplied by Windows:

Image **The Windows Biometric Service (%SystemRoot%\System32\Wbiosrvc.dll** This provides the process-execution environment in which one or more biometric service providers can execute.

Image **The Windows Biometric Driver Interface (WBDI)** This is a set of interface definitions (IRP major function codes, DeviceIoControl codes, and so forth) to which any driver for a biometric scanner device must conform if it is to be compatible with the Windows Biometric Service. WBDI drivers can be developed using any of the standard driver frameworks (UMDF, KMDF and WDM). However, UMDF is recommended to reduce code size and increase reliability. WBDI is described in the Windows Driver Kit documentation.

Image **The Windows Biometric API** This allows existing Windows components such as Winlogon and LogonUI to access the biometric service. Third-party applications have access to the Windows Biometric API and can use the biometric scanner for functions other than logging in to Windows. An example of a function in this API is WinBioEnumServiceProviders. The Biometric API is exposed by %SystemRoot%\System32\Winbio.dll.

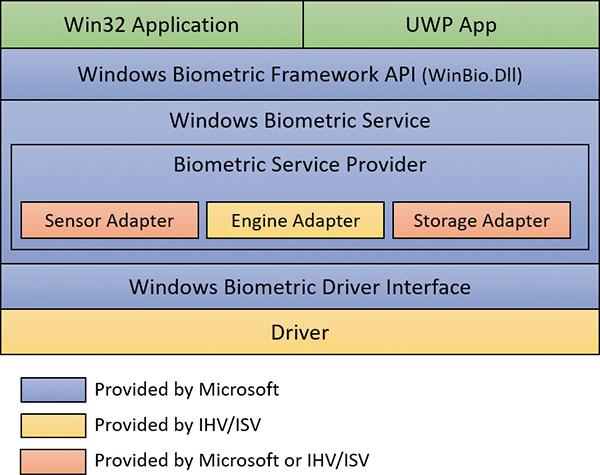
Image **The fingerprint biometric service provider** This wraps the functions of biometric-type-specific adapters to present a common interface, independent of the type of biometric, to the Windows Biometric Service. In the future, additional types of biometrics, such as retinal scans or voiceprint analyzers, might be supported by additional biometric service providers. The biometric service provider in turn uses three adapters, which are user-mode DLLs:

• **The sensor adapter** This exposes the data-capture functionality of the scanner. The sensor adapter usually uses Windows I/O calls to access the scanner hardware. Windows provides a sensor adapter that can be used with simple sensors, those for which a WBDI driver exists. For more complex sensors, the sensor adapter is written by the sensor vendor.

• **The engine adapter** This exposes processing and comparison functionality specific to the scanner’s raw data format and other features. The actual processing and comparison might be performed within the engine adapter DLL, or the DLL might communicate with some other module. The engine adapter is always provided by the sensor vendor.

• **The storage adapter** This exposes a set of secure storage functions. These are used to store and retrieve templates against which scanned biometric data is matched by the engine adapter. Windows provides a storage adapter using Windows cryptography services and standard disk file storage. A sensor vendor might provide a different storage adapter.

Image **The functional device driver for the actual biometric scanner device** This exposes the WBDI at its upper edge. It usually uses the services of a lower-level bus driver, such as the USB bus driver, to access the scanner device. This driver is always provided by the sensor vendor.



**FIGURE 7-21** Windows Biometric Framework components and architecture.

A typical sequence of operations to support logging in via a fingerprint scan might be as follows:

**1.** After initialization, the sensor adapter receives from the service provider a request for capture data. The sensor adapter in turn sends a DeviceIoControl request with the IOCTL\_BIOMETRIC\_CAPTURE\_DATA control code to the WBDI driver for the fingerprint scanner device.

**2.** The WBDI driver puts the scanner into capture mode and queues the IOCTL\_BIOMETRIC\_CAPTURE\_DATA request until a fingerprint scan occurs.

**3.** A prospective user swipes a finger across the scanner. The WBDI driver receives notification of this, obtains the raw scan data from the sensor, and returns this data to the sensor driver in a buffer associated with the IOCTL\_BIOMETRIC\_CAPTURE\_DATA request.

**4.** The sensor adapter provides the data to the fingerprint biometric service provider, which in turn passes the data to the engine adapter.

**5.** The engine adapter processes the raw data into a form compatible with its template storage.

**6.** The fingerprint biometric service provider uses the storage adapter to obtain templates and corresponding security IDs from secure storage. It invokes the engine adapter to compare each template to the processed scan data. The engine adapter returns a status indicating whether it’s a match or not a match.

**7.** If a match is found, the Windows Biometric Service notifies Winlogon, via a credential provider DLL, of a successful login and passes it the security ID of the identified user. This notification is sent via an ALPC message, providing a path that cannot be spoofed.

#### **Windows Hello**

Windows Hello, introduced in Windows 10, provides new ways to authenticate users based on biometric information. With this technology, users can log in effortlessly just by showing themselves to the device’s camera or swiping their finger.

At the time of this writing, Windows Hello supports three types of biometric identification:

Image Fingerprint

Image Face

Image Iris

The security aspect of biometrics needs to be considered first. What is the likelihood of someone being identified as you? What is the likelihood of you *not* being identified as you? These questions are parameterized by two factors:

Image **False accept rate (uniqueness)** This is the probability of another user having the same biometric data as you. Microsoft’s algorithms make sure the likelihood is 1 in 100,000.

Image **False reject rate (reliability)** This is the probability of you not being correctly recognized as you (for example, in abnormal lighting conditions for face or iris recognition). Microsoft’s implementation makes sure there is less than 1 percent chance of this happening. If it does happen, the user can try again or use a PIN code instead.

Using a PIN code may seem less secure than using a full-blown password (the PIN can be as simple as a four-digit number). However, a PIN is more secure than a password for two main reasons:

Image The PIN code is local to the device and is never transmitted across the network. This means that even if someone gets a hold of the PIN, they cannot use it to log in as the user from any other device. Passwords, on the other hand, travel to the domain controller. If someone gets hold of the password, they can log in from another machine into the domain.

Image The PIN code is stored in the Trusted Platform Module (TPM)—a piece of hardware that also plays a part in Secure Boot (discussed in detail in Chapter 11 in Part 2)—so is difficult to access. In any case, it requires physical access to the device, raising the bar considerably for a potential security compromise.

Windows Hello is built upon the Windows Biometric Framework (WBF) (described in the previous section). Current laptop devices support fingerprint and face biometrics, while iris is only supported on the Microsoft Lumia 950 and 950 XL phones. (This will likely change and expand in future devices.) Note that face recognition requires an infrared (IR) camera as well as a normal (RGB) one, and is supported on devices such as the Microsoft Surface Pro 4 and the Surface Book.

### **USER ACCOUNT CONTROL AND VIRTUALIZATION**

User Account Control (UAC) is meant to enable users to run with standard user rights as opposed to administrative rights. Without administrative rights, users cannot accidentally (or deliberately) modify system settings, malware can’t normally alter system security settings or disable antivirus software, and users can’t compromise the sensitive information of other users on shared computers. Running with standard user rights can thus mitigate the impact of malware and protect sensitive data on shared computers.

UAC had to address a couple of problems to make it practical for a user to run with a standard user account. First, because the Windows usage model has been one of assumed administrative rights, software developers assumed their programs would run with those rights and could therefore access and modify any file, registry key, or operating system setting. Second, users sometimes need administrative rights to perform such operations as installing software, changing the system time, and opening ports in the firewall.

The UAC solution to these problems is to run most applications with standard user rights, even though the user is logged in to an account with administrative rights. At the same time, UAC makes it possible for standard users to access administrative rights when they need them—whether for legacy applications that require them or for changing certain system settings. As described, UAC accomplishes this by creating a filtered admin token as well as the normal admin token when a user logs in to an administrative account. All processes created under the user’s session will normally have the filtered admin token in effect so that applications that can run with standard user rights will do so. However, the administrative user can run a program or perform other functions that require full Administrator rights through UAC elevation.

Windows also allows certain tasks that were previously reserved for administrators to be performed by standard users, enhancing the usability of the standard user environment. For example, Group Policy settings exist that can enable standard users to install printers and other device drivers approved by IT administrators and to install ActiveX controls from administrator-approved sites.

Finally, when software developers test in the UAC environment, they are encouraged to develop applications that can run without administrative rights. Fundamentally, non-administrative programs should not need to run with administrator privileges; programs that often require administrator privileges are typically legacy programs using old APIs or techniques, and they should be updated.

Together, these changes obviate the need for users to run with administrative rights all the time.

#### **File system and registry virtualization**

Although some software legitimately requires administrative rights, many programs needlessly store user data in system-global locations. When an application executes, it can be running in different user accounts, and it should therefore store user-specific data in the per-user %AppData% directory and save per-user settings in the user’s registry profile under HKEY\_CURRENT\_USER\Software. Standard user accounts don’t have write access to the %ProgramFiles% directory or HKEY\_LOCAL\_MACHINE\Software, but because most Windows systems are single-user and most users have been administrators until UAC was implemented, applications that incorrectly saved user data and settings to these locations worked anyway.

Windows enables these legacy applications to run in standard user accounts through the help of file system and registry namespace virtualization. When an application modifies a system-global location in the file system or registry and that operation fails because access is denied, Windows redirects the operation to a per-user area. When the application reads from a system-global location, Windows first checks for data in the per-user area and, if none is found, permits the read attempt from the global location.

Windows will always enable this type of virtualization unless:

Image **The application is 64-bit** Because virtualization is purely an application-compatibility technology meant to help legacy applications, it is enabled only for 32-bit applications. The world of 64-bit applications is relatively new and developers should follow the development guidelines for creating standard user-compatible applications.

Image **The application is already running with administrative rights** In this case, there is no need for any virtualization.

Image **The operation came from a kernel-mode caller**

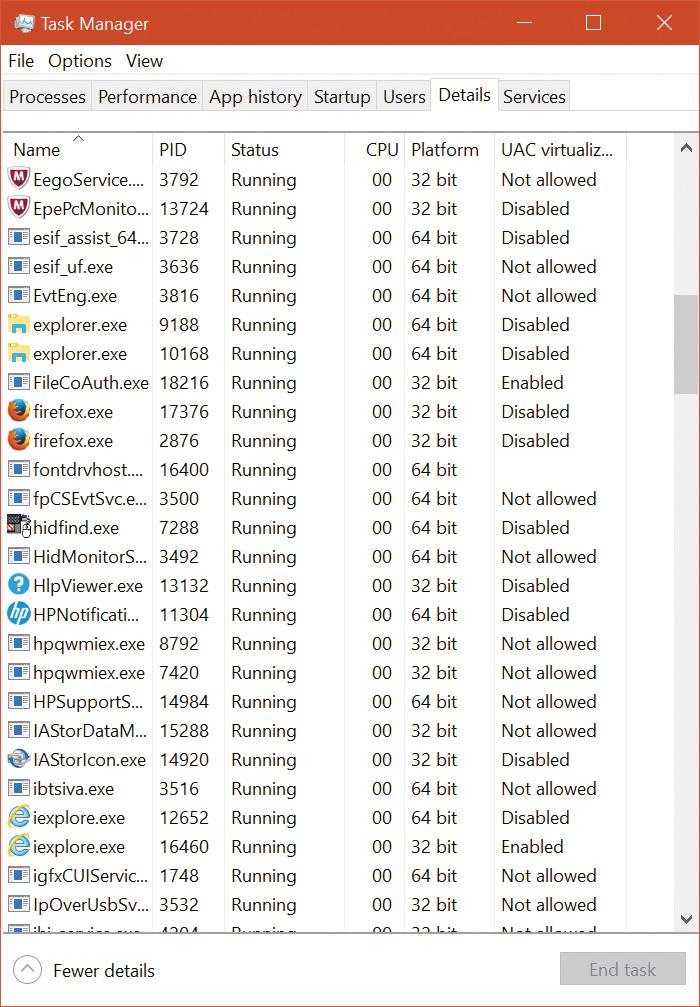
**Image** **The operation is being performed while the caller is impersonating** For example, any operations not originating from a process classified as legacy according to this definition, including network file-sharing accesses, are not virtualized.

Image **The executable image for the process has a UAC-compatible manifest** Specifying a requestedExecutionLevel setting, described in the next section.

Image **The administrator does not have write access to the file or registry key** This exception exists to enforce backward compatibility because the legacy application would have failed before UAC was implemented even if the application was run with administrative rights.

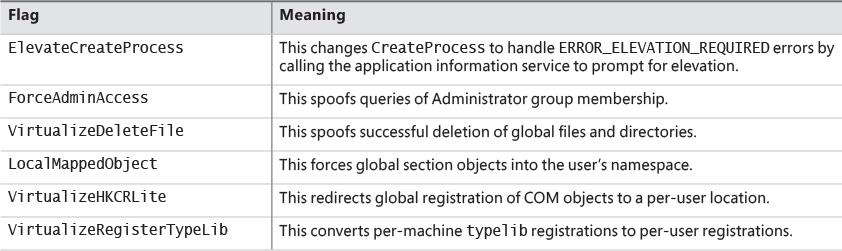
Image **Services are never virtualized**

You can see the virtualization status (the process virtualization status is stored as a flag in its token) of a process by adding the UAC Virtualization column to Task Manager’s Details page, as shown in [Figure 7-22](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig22). Most Windows components—including the Desktop Window Manager (Dwm.exe), the Client Server Run-Time Subsystem (Csrss.exe), and Explorer—have virtualization disabled because they have a UAC-compatible manifest or are running with administrative rights and so do not allow virtualization. However, 32-bit Internet Explorer (iexplore.exe) has virtualization enabled because it can host multiple ActiveX controls and scripts and must assume that they were not written to operate correctly with standard user rights. Note that, if required, virtualization can be completely disabled for a system using a Local Security Policy setting.



**FIGURE 7-22** Using Task Manager to view virtualization status.

In addition to file system and registry virtualization, some applications require additional help to run correctly with standard user rights. For example, an application that tests the account in which it’s running for membership in the Administrators group might otherwise work, but it won’t run if it’s not in that group. Windows defines a number of application-compatibility shims to enable such applications to work anyway. The shims most commonly applied to legacy applications for operation with standard user rights are shown in [Table 7-15](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07tab15).



**TABLE 7-15** UAC virtualization shims

##### **File virtualization**

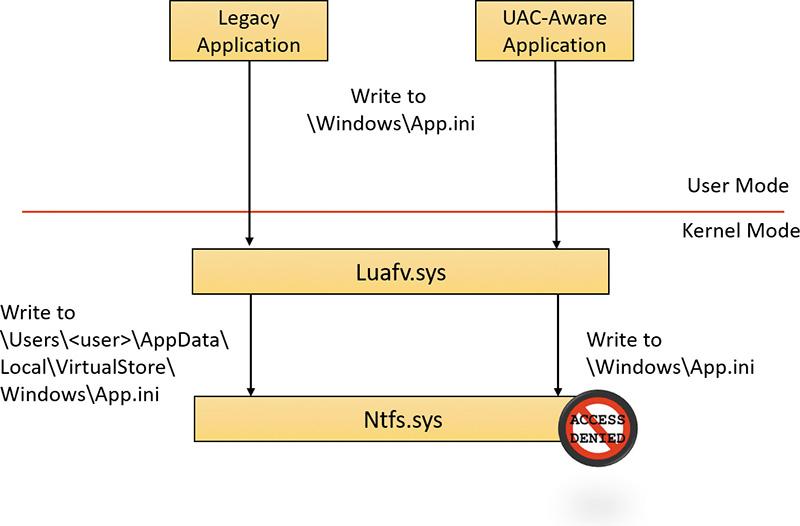
The file system locations that are virtualized for legacy processes are %ProgramFiles%, %ProgramData%, and %SystemRoot%, excluding some specific subdirectories. However, any file with an executable extension—including .exe, .bat, .scr, .vbs, and others—is excluded from virtualization. This means that programs that update themselves from a standard user account fail instead of creating private versions of their executables that aren’t visible to an administrator running a global updater.

Image Note

To add extensions to the exception list, enter them in the HKLM\System\Current-ControlSet\Services\Luafv\Parameters\ExcludedExtensionsAdd registry key and reboot. Use a multistring type to delimit multiple extensions, and do not include a leading dot in the extension name.

Modifications to virtualized directories by legacy processes are redirected to the user’s virtual root directory, %LocalAppData%\VirtualStore. The Local component of the path highlights the fact that virtualized files don’t roam with the rest of the profile when the account has a roaming profile.

The UAC File Virtualization filter driver (%SystemRoot%\System32\Drivers\Luafv.sys) implements file system virtualization. Because this is a file system filter driver, it sees all local file system operations, but it implements functionality only for operations from legacy processes. As shown in [Figure 7-23](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig23), the filter driver changes the target file path for a legacy process that creates a file in a system-global location but does not for a non-virtualized process with standard user rights. Default permissions on the \Windows directory deny access to the application written with UAC support, but the legacy process acts as though the operation succeeds when it really created the file in a location fully accessible by the user.



**FIGURE 7-23** UAC File Virtualization filter driver operation.

EXPERIMENT: File virtualization behavior

In this experiment, you will enable and disable virtualization on the command prompt and see several behaviors to demonstrate UAC file virtualization:

**1.** Open a non-elevated command prompt (you must have UAC enabled for this to work) and enable virtualization for it. You can change the virtualization status of a process by right-clicking the process in the Task Manager Details tab and selecting **UAC Virtualization** from the shortcut menu that appears.

**2.** Navigate to the C:\Windows directory and use the following command to write a file:

echo hello-1 > test.txt

**3.** List the contents of the directory. You’ll see that the file appears.

dir test.txt

**4.** Disable virtualization by right-clicking the process in the Task Manager Details tab and deselecting **UAC Virtualization**. Then list the directory as in step 3. Notice that the file is gone. However, a directory listing of the VirtualStore directory will reveal the file:

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p726pro01a)

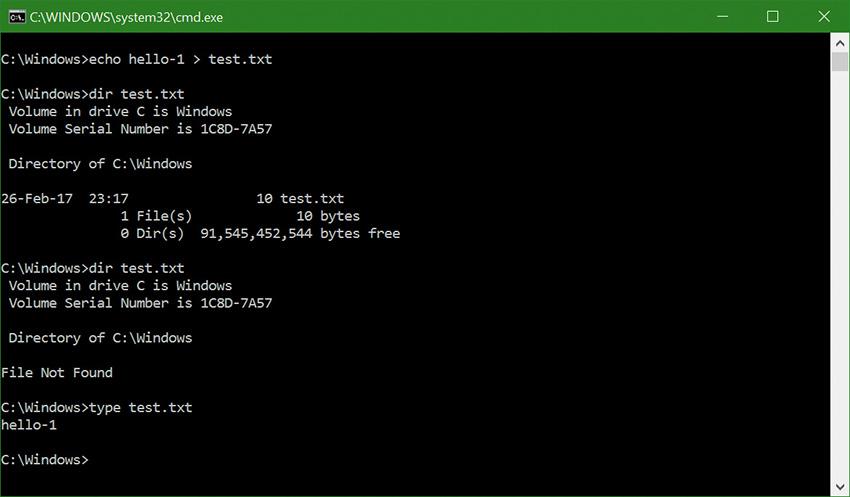
dir %LOCALAPPDATA%\VirtualStore\Windows\test.txt

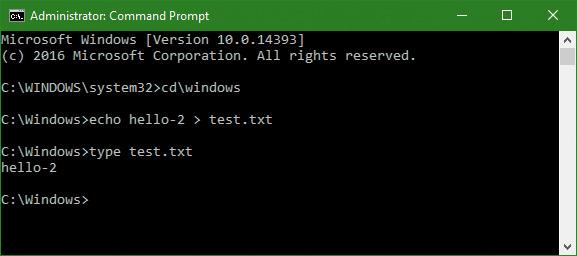
**5.** Enable virtualization again for this process.

**6.** To look at a more complex scenario, create a new command prompt window, but elevated this time. Then repeat steps 2 and 3 using the string hello-2.

**7.** Examine the text inside these files by using the following command in both command prompts. The screenshots that follow show the expected output.

type test.txt





**8.** From your elevated command prompt, delete the test.txt file:

del test.txt

**9.** Repeat step 3 of the experiment in both windows. Notice that the elevated command prompt cannot find the file anymore, while the standard user command prompt shows the old contents of the file again. This demonstrates the failover mechanism described earlier: Read operations look in the per-user virtual store location first, but if the file doesn’t exist, read access to the system location will be granted.

##### **Registry virtualization**

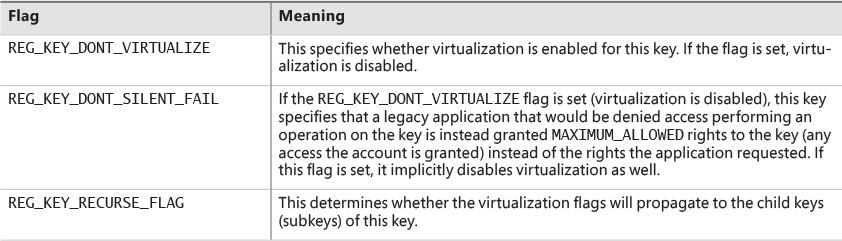
Registry virtualization is implemented slightly differently from file system virtualization. Virtualized registry keys include most of the HKEY\_LOCAL\_MACHINE\Software branch, but there are numerous exceptions, such as the following:

Image HKLM\Software\Microsoft\Windows

Image HKLM\Software\Microsoft\Windows NT

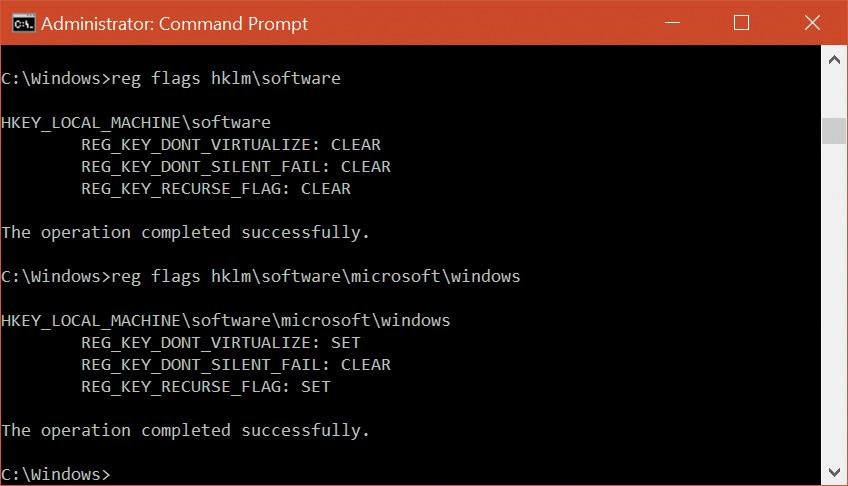
Image HKLM\Software\Classes

Only keys that are commonly modified by legacy applications, but that don’t introduce compatibility or interoperability problems, are virtualized. Windows redirects modifications of virtualized keys by a legacy application to a user’s registry virtual root at HKEY\_CURRENT\_USER\Software\Classes\VirtualStore. The key is located in the user’s Classes hive, %LocalAppData%\Microsoft\Windows\UsrClass.dat, which, like any other virtualized file data, does not roam with a roaming user profile. Instead of maintaining a fixed list of virtualized locations as Windows does for the file system, the virtualization status of a key is stored as a combination of flags, shown in [Table 7-16](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07tab16).



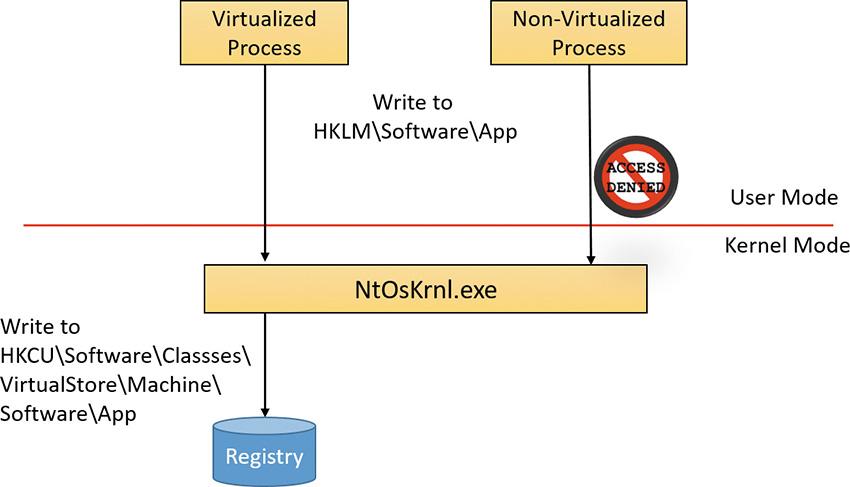
**TABLE 7-16** Registry virtualization flags

You can use the Reg.exe utility included in Windows, with the flags option, to display the current virtualization state for a key or to set it. In [Figure 7-24](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig24), note that the HKLM\Software key is fully virtualized, but the Windows subkey (and all its children) have only silent failure enabled.



**FIGURE 7-24** UAC registry virtualization flags on the Software and Windows keys.

Unlike file virtualization, which uses a filter driver, registry virtualization is implemented in the configuration manager. (See Chapter 9 in Part 2 for more information on the registry and the configuration manager.) As with file system virtualization, a legacy process creating a subkey of a virtualized key is redirected to the user’s registry virtual root, but a UAC-compatible process is denied access by default permissions. This is shown in [Figure 7-25](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig25).



**FIGURE 7-25** UAC registry virtualization operation.

#### **Elevation**

Even if users run only programs that are compatible with standard user rights, some operations still require administrative rights. For example, the vast majority of software installations require administrative rights to create directories and registry keys in system-global locations or to install services or device drivers. Modifying system-global Windows and application settings also requires administrative rights, as does the parental controls feature. It would be possible to perform most of these operations by switching to a dedicated administrator account, but the inconvenience of doing so would likely result in most users remaining in the administrator account to perform their daily tasks, most of which do not require administrative rights.

It’s important to be aware that UAC elevations are conveniences and not security boundaries. A security boundary requires that security policy dictate what can pass through the boundary. User accounts are an example of a security boundary in Windows because one user can’t access the data belonging to another user without having that user’s permission.

Because elevations aren’t security boundaries, there’s no guarantee that malware running on a system with standard user rights can’t compromise an elevated process to gain administrative rights. For example, elevation dialog boxes only identify the executable that will be elevated; they say nothing about what it will do when it executes.

##### **Running with administrative rights**

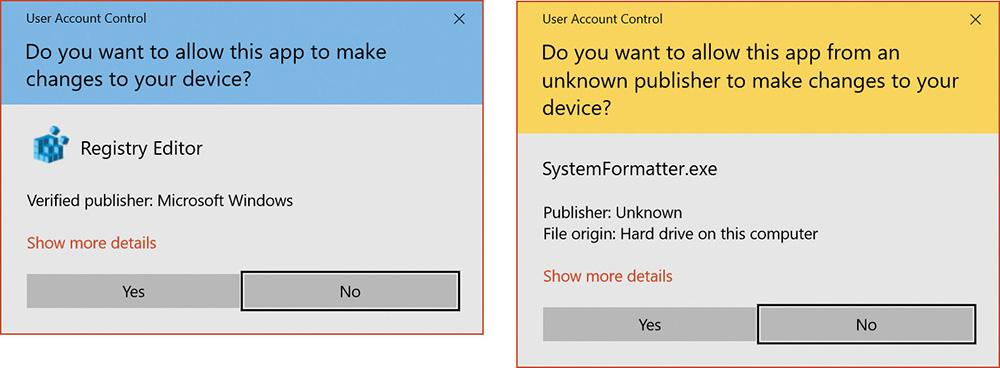
Windows includes enhanced “run as” functionality so that standard users can conveniently launch processes with administrative rights. This functionality requires giving applications a way to identify operations for which the system can obtain administrative rights on behalf of the application, as necessary (we’ll say more on this topic shortly).

To enable users acting as system administrators to run with standard user rights but not have to enter user names and passwords every time they want to access administrative rights, Windows makes use of a mechanism called *Admin Approval Mode* (*AAM*). This feature creates two identities for the user at logon: one with standard user rights and another with administrative rights. Since every user on a Windows system is either a standard user or acting for the most part as a standard user in AAM, developers must assume that all Windows users are standard users, which will result in more programs working with standard user rights without virtualization or shims.

Granting administrative rights to a process is called *elevation*. When elevation is performed by a standard user account (or by a user who is part of an administrative group but not the actual Administrators group), it’s referred to as an *over-the-shoulder* (*OTS*) *elevation* because it requires the entry of credentials for an account that’s a member of the Administrators group, something that’s usually completed by a privileged user typing over the shoulder of a standard user. An elevation performed by an AAM user is called a *consent elevation* because the user simply has to approve the assignment of his administrative rights.

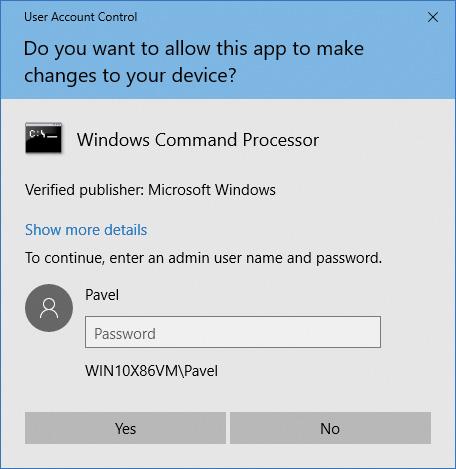
Stand-alone systems, which are typically home computers, and domain-joined systems treat AAM access by remote users differently because domain-connected computers can use domain administrative groups in their resource permissions. When a user accesses a stand-alone computer’s file share, Windows requests the remote user’s standard user identity. But on domain-joined systems, Windows honors all the user’s domain group memberships by requesting the user’s administrative identity. Executing an image that requests administrative rights causes the application information service (AIS, contained in %SystemRoot%\System32\Appinfo.dll), which runs inside a standard service host process (SvcHost.exe), to launch %SystemRoot%\System32\Consent.exe. Consent captures a bitmap of the screen, applies a fade effect to it, switches to a desktop that’s accessible only to the local system account (the Secure Desktop), paints the bitmap as the background, and displays an elevation dialog box that contains information about the executable. Displaying this dialog box on a separate desktop prevents any application present in the user’s account from modifying the appearance of the dialog box.

If an image is a Windows component digitally signed (by Microsoft or another entity), the dialog box displays a light blue stripe across the top, as shown at the left of [Figure 7-26](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig26) (the distinction between Microsoft signed images and other signers has been removed in Windows 10). If the image is unsigned, the stripe becomes yellow, and the prompt stresses the unknown origin of the image (see the right of [Figure 7-26](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig26)). The elevation dialog box shows the image’s icon, description, and publisher for digitally signed images, but it shows only the file name and “Publisher: Unknown” for unsigned images. This difference makes it harder for malware to mimic the appearance of legitimate software. The Show More Details link at the bottom of the dialog box expands it to show the command line that will be passed to the executable if it launches.



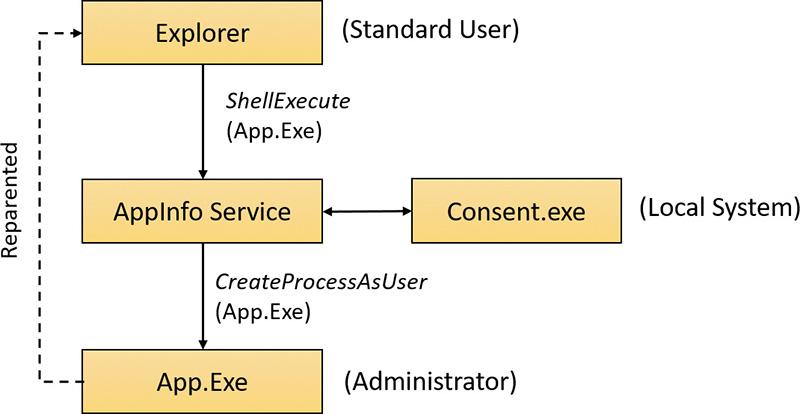
**FIGURE 7-26** AAC UAC elevation dialog boxes based on image signature.

The OTS consent dialog box, shown in [Figure 7-27](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig27), is similar, but prompts for administrator credentials. It will list any accounts with administrator rights.



**FIGURE 7-27** OTS consent dialog box.

If a user declines an elevation, Windows returns an access-denied error to the process that initiated the launch. When a user agrees to an elevation by either entering administrator credentials or clicking Yes, AIS calls CreateProcessAsUser to launch the process with the appropriate administrative identity. Although AIS is technically the parent of the elevated process, AIS uses new support in the CreateProcessAsUser API that sets the process’s parent process ID to that of the process that originally launched it. That’s why elevated processes don’t appear as children of the AIS service-hosting process in tools such as Process Explorer that show process trees. [Figure 7-28](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig28) shows the operations involved in launching an elevated process from a standard user account.



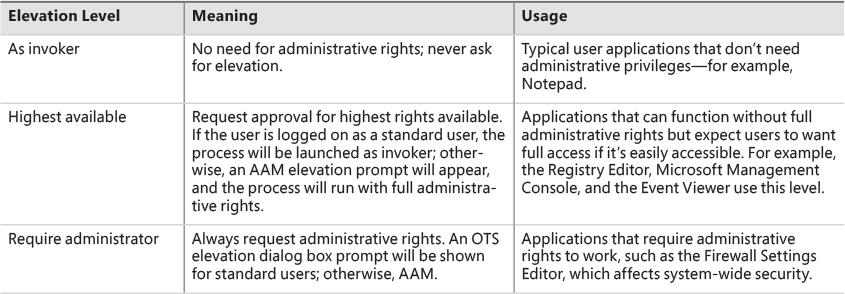
**FIGURE 7-28** Launching an administrative application as a standard user.

##### **Requesting administrative rights**

There are a number of ways the system and applications identify a need for administrative rights. One that shows up in the Explorer user interface is the Run as Administrator context menu command and shortcut option. These items also include a blue and gold shield icon that should be placed next to any button or menu item that will result in an elevation of rights when it is selected. Choosing the Run as Administrator command causes Explorer to call the ShellExecute API with the runas verb.

The vast majority of installation programs require administrative rights, so the image loader, which initiates the launch of an executable, includes installer-detection code to identify likely legacy installers. Some of the heuristics it uses are as simple as detecting internal version information or whether the image has the words *setup*, *install*, or *update* in its file name. More sophisticated means of detection involve scanning for byte sequences in the executable that are common to third-party installation wrapper utilities. The image loader also calls the application-compatibility library to see if the target executable requires administrator rights. The library looks in the application-compatibility database to see whether the executable has the RequireAdministrator or RunAsInvoker compatibility flag associated with it.

The most common way for an executable to request administrative rights is for it to include a requestedExecutionLevel tag in its application manifest file. The element’s level attribute can have one of the three values shown in [Table 7-17](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07tab17).



**TABLE 7-17** Requested elevation levels

The presence of the trustInfo element in a manifest (which you can see in the manifest dump of eventvwr.exe) denotes an executable that was written with support for UAC and the requestedExecutionLevel element nests within it. The uiAccess attribute is where accessibility applications can use the UIPI bypass functionality mentioned earlier.

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p732pro01a)

C:\>sigcheck -m c:\Windows\System32\eventvwr.exe

...

<trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">

<security>

<requestedPrivileges>

<requestedExecutionLevel

level="highestAvailable"

uiAccess="false"

/>

</requestedPrivileges>

</security>

</trustInfo>

<asmv3:application>

<asmv3:windowsSettings xmlns="http://schemas.microsoft.com/SMI/2005/WindowsSettings">

<autoElevate>true</autoElevate>

</asmv3:windowsSettings>

</asmv3:application>

...

##### **Auto-elevation**

In the default configuration (see the next section for information on changing this), most Windows executables and control panel applets do not result in elevation prompts for administrative users, even if they need administrative rights to run. This is because of a mechanism called *auto-elevation*. Auto-elevation is intended to preclude administrative users from seeing elevation prompts for most of their work; the programs will automatically run under the user’s full administrative token.

Auto-elevation has several requirements. One is that the executable in question must be considered as a Windows executable. This means it must be signed by the Windows publisher (not just by Microsoft; oddly, they are not the same—Windows-signed is considered more privileged than Microsoft-signed). It must also be in one of several directories considered secure: %SystemRoot%\System32 and most of its subdirectories, %Systemroot%\Ehome, and a small number of directories under %ProgramFiles% (for example, those containing Windows Defender and Windows Journal).

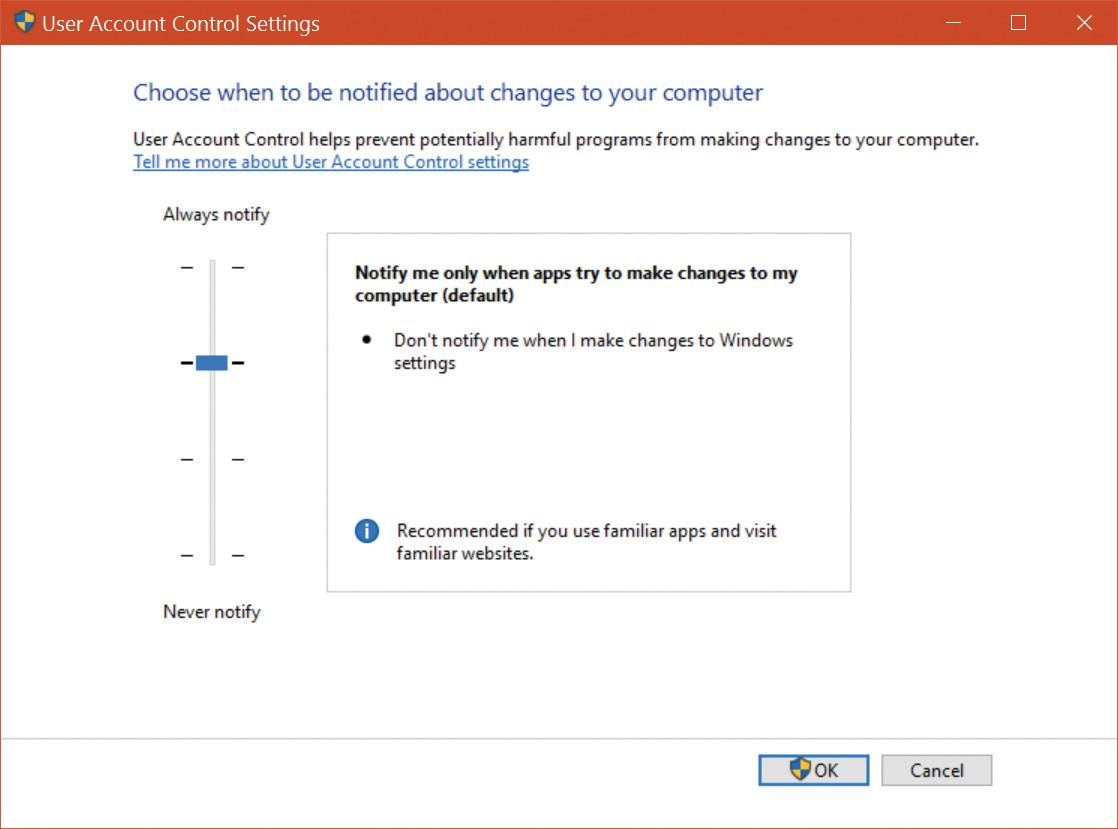
There are additional requirements, depending on the type of executable. EXE files other than Mmc.exe auto-elevate if they are requested via an autoElevate element in their manifest. The manifest shown earlier of eventvwr.exe in the previous section illustrates this.

Mmc.exe is treated as a special case because whether it should auto-elevate or not depends on which system management snap-ins it is to load. Mmc.exe is normally invoked with a command line specifying an MSC file, which in turn specifies which snap-ins are to be loaded. When Mmc.exe is run from a protected administrator account (one running with the limited administrator token), it asks Windows for administrative rights. Windows validates that Mmc.exe is a Windows executable and then checks the MSC. The MSC must also pass the tests for a Windows executable, and furthermore must be on an internal list of auto-elevate MSCs. This list includes nearly all MSC files in Windows.

Finally, COM (out-of-process server) classes can request administrative rights within their registry key. To do so requires a subkey named Elevation with a DWORD value named Enabled, having a value of 1. Both the COM class and its instantiating executable must meet the Windows executable requirements, although the executable need not have requested auto-elevation.

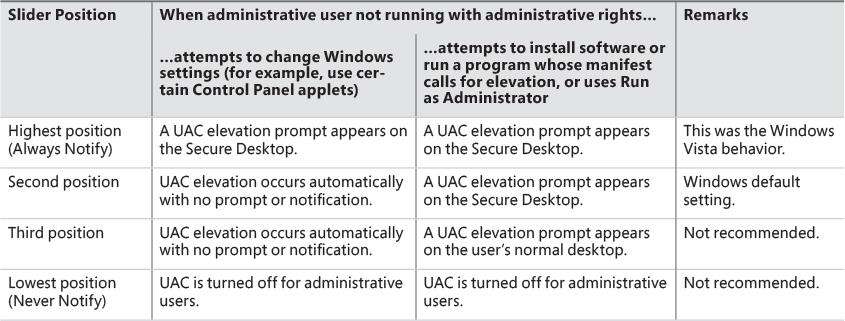
##### **Controlling UAC behavior**

UAC can be modified via the dialog box shown in [Figure 7-29](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig29). This dialog box is available under Change User Account Control Settings. [Figure 7-29](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig29) shows the control in its default position.



**FIGURE 7-29** User Account Control Settings dialog box.

The four possible settings have the effects described in [Table 7-18](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07tab18).

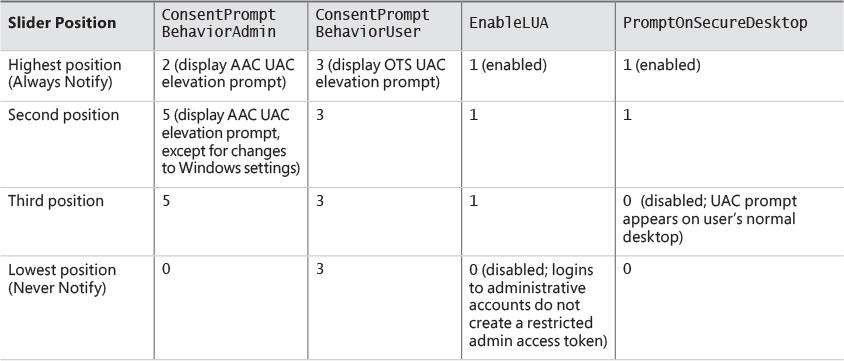


**TABLE 7-18** UAC options

The third position is not recommended because the UAC elevation prompt appears not on the Secure Desktop but on the normal user’s desktop. This could allow a malicious program running in the same session to change the appearance of the prompt. It is intended for use only in systems where the video subsystem takes a long time to dim the desktop or is otherwise unsuitable for the usual UAC display.

The lowest position is strongly discouraged because it turns UAC off completely as far as administrative accounts are concerned. Prior to Windows 8, all processes run by a user with an administrative account are run with the user’s full administrative rights in effect; there is no filtered admin token. Starting with Windows 8, UAC cannot be turned off completely, because of the AppContainer model. Admin users won’t be prompted for elevation, but processes will not elevate unless required to do so by the manifest or launched from an elevated process.

The UAC setting is stored in four values in the registry under HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System, as shown in [Table 7-19](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07tab19). ConsentPromptBehaviorAdmin controls the UAC elevation prompt for administrators running with a filtered admin token, and ConsentPromptBehaviorUser controls the UAC prompt for users other than administrators.



**TABLE 7-19** UAC registry values

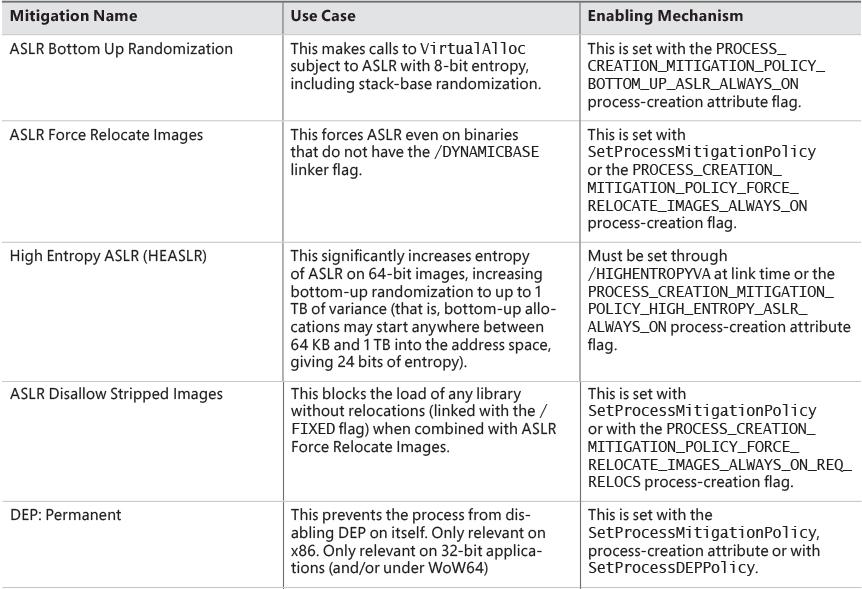
### **EXPLOIT MITIGATIONS**

Throughout this chapter, we’ve seen a number of technologies that help protect the user, guarantee the code-signing properties of executable code, and lock down access to resources through sandboxing. At the end of the day, however, all secure systems have failure points, all code has bugs, and attackers leverage increasingly complex attacks to exploit them. A security model in which all code is assumed to be bug-free, or in which a software developer assumes all bugs will eventually be found and fixed, is destined to fail. Additionally, many security features that provide code-execution “guarantees” do so at a cost of performance or compatibility, which may be unacceptable in such scenarios.

A much more successful approach is to identify the most common techniques used by attackers, as well as employ an internal “red team” (that is, an internal team attacking its own software) to discover new techniques before attackers do and to implement mitigations against such techniques. (These mitigations can be as simple as moving some data around or as complex as employing Control Flow Integrity [CFI] techniques.) Because vulnerabilities can number in the thousands in a complex code base such as Windows, but exploit techniques are limited, the idea is to make large classes of bugs very difficult (or in some cases, impossible) to exploit, without worrying about finding all the bugs.

#### **Process-mitigation policies**

While individual applications can implement various exploit mitigations on their own (such as Microsoft Edge, which leverages a mitigation called *MemGC* to avoid many classes of memory-corruption attacks), this section will cover mitigations that are provided by the operating system to all applications or to the system itself to reduce exploitable bug classes. [Table 7-20](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07tab20) describes all mitigations in the latest version of Windows 10 Creators Update, the type of bug class they mitigate against, and mechanisms to activate them.



Image

Image

Image

Image

Image

**TABLE 7-20** Process mitigation options

Note that it is also possible to some of these mitigations on a per-application or per-system basis without the cooperation of the application developer. To do so, open the Local Group Policy Editor. Then expand **Computer Configuration**, then **Administrative Templates**, then **System**, and finally **Mitigation Options** (see [Figure 7-30](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig30)). In the Process Mitigation Options dialog box, enter the appropriate bit-number value that corresponds to the mitigations being enabled, using **1** to enable a mitigation, **0** to disable it, or **?** to leave it to its default or process-requested value (again, see [Figure 7-30](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig30)). The bit numbers are taken from the PROCESS\_MITIGATION\_POLICY enumeration found in the Winnt.h header file. This will result in the appropriate registry value being written in the Image File Execution Options (IFEO) key for the entered image name. Unfortunately the current version of Windows 10 Creators Update and earlier will strip out many of the newer mitigations. You can avoid this by manually setting the REG\_DWORD MitigationOptions registry value.

Image

**FIGURE 7-30** Customizing process-mitigation options.

#### **Control Flow Integrity**

Data Execution Prevention (DEP) and Arbitrary Code Guard (ACG) make it hard for exploits to place executable code on the heap or stack, to allocate new executable code, or to change existing executable code. As a result, memory/data-only attacks have become more interesting. Such attacks allow the modification of portions of memory to redirect control flow, such as modifying return addresses on the stack or indirect function pointers stored in memory. Techniques such as return-oriented-programming (ROP) and jump-oriented-programming (JOP) are often used to violate the regular code flow of the program and redirect it to known locations of interesting code snippets (“gadgets”).

Because such snippets are often present in the middle or end of various functions, when control flow is redirected in this way, it must be redirected into the middle or end of a legitimate function. By employing Control Flow Integrity (CFI) technologies—which can, for example, validate that the target of an indirect JMP or CALL instruction is the beginning of a real function, or that a RET instruction is pointing to an expected location, or that a RET instruction is issued after the function was entered through its beginning—the operating system and compiler can detect and prevent most classes of such exploits.

##### **Control Flow Guard**

Control Flow Guard (CFG) is an exploit-mitigation mechanism first introduced in Windows 8.1 Update 3 that exists in enhanced version in Windows 10 and Server 2016, with further improvements released on various updates (up to and including the latest Creators Update). Originally implemented only for user-mode code, CFG now also exists as Kernel CFG (KCFG) on the Creators Update. CFG addresses the indirect CALL/JMP part of CFI by verifying that the target of an indirect call is at the start of a known function (more on that momentarily). If the target is not at the start of a known function, the process is simply terminated. [Figure 7-31](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig31) shows the conceptual operation of CFG.

Image

**FIGURE 7-31** Conceptual view of Control Flow Guard.

CFG requires the cooperation of a supported compiler that will add the call to the validation code before indirect changes in control flow. The Visual C++ compiler has an option, /guard:cf, that must be set for images (or even on a C/C++ source file level) to be built with CFG support (this option is also available in Visual Studio’s GUI in the C/C++/Code Generation/Control Flow Guard setting in the project’s properties). This setting should also be set in the linker settings, as both components of Visual Studio are required to collaborate to support CFG.

Once those settings are present, images (EXEs and DLLs) that are compiled with CFG-enabled indicate this in their PE header. In addition, they contain a list of functions that are the valid indirect control flow targets in a .gfids PE section (by default merged by the linker with the .rdata section). This list is built by the linker and contains the relative virtual address (RVA) of all functions in the image. This includes those that might not be called by an indirect call by the code present in the image because there’s no way of knowing if outside code does not somehow legitimately know the address of a function and is attempting to call it. This can be especially true of exported functions, which can be called after obtaining their pointer through GetProcAddress.

That being said, programmers can use a technique called *CFG suppression*, which is supported through the DECLSPEC\_GUARD\_SUPRESS annotation, and which marks the function in the table of valid functions with a special flag indicating that the programmer never expects such a function to be the target of any indirect call or jump.

Now that a table of valid function targets exists, all that a simple validation function would need to do is to compare the target of the CALL or JMP instruction with one of the functions in the table. Algorithmically, this would result in an O(*n*) algorithm, where the number of functions needed to check would be equivalent, in the worst case, to the number of functions in the table. Clearly, linearly scanning an entire array during every single indirect change in control flow would bring a program to its knees, so operating system support is needed to perform CFG checks efficiently. We’ll see in the next section how Windows can achieve this.

EXPERIMENT: Control Flow Guard information

The DumpBin Visual Studio tool can show some basic CFG information. The following dumps header and loader configuration information for Smss:

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p742pro01a)

c:\> dumpbin /headers /loadconfig c:\windows\system32\smss.exe

Microsoft (R) COFF/PE Dumper Version 14.00.24215.1

Copyright (C) Microsoft Corporation. All rights reserved.

Dump of file c:\windows\system32\smss.exe

PE signature found

File Type: EXECUTABLE IMAGE

FILE HEADER VALUES

8664 machine (x64)

6 number of sections

57899A7D time date stamp Sat Jul 16 05:22:53 2016

0 file pointer to symbol table

0 number of symbols

F0 size of optional header

22 characteristics

Executable

Application can handle large (>2GB) addresses

OPTIONAL HEADER VALUES

20B magic # (PE32+)

14.00 linker version

12800 size of code

EC00 size of initialized data

0 size of uninitialized data

1080 entry point (0000000140001080) NtProcessStartupW

1000 base of code

140000000 image base (0000000140000000 to 0000000140024FFF)

1000 section alignment

200 file alignment

10.00 operating system version

10.00 image version

10.00 subsystem version

0 Win32 version

25000 size of image

400 size of headers

270FD checksum

1 subsystem (Native)

4160 DLL characteristics

High Entropy Virtual Addresses

Dynamic base

NX compatible

**Control Flow Guard**

...

Section contains the following load config:

000000D0 size

0 time date stamp

0.00 Version

0 GlobalFlags Clear

0 GlobalFlags Set

0 Critical Section Default Timeout

0 Decommit Free Block Threshold

0 Decommit Total Free Threshold

0000000000000000 Lock Prefix Table

0 Maximum Allocation Size

0 Virtual Memory Threshold

0 Process Heap Flags

0 Process Affinity Mask

0 CSD Version

0800 Dependent Load Flag

0000000000000000 Edit List

0000000140020660 Security Cookie

**00000001400151C0 Guard CF address of check-function pointer**

**00000001400151C8** **Guard CF address of dispatch-function pointer**

00000001400151D0 **Guard CF function table**

2A **Guard CF function count**

00010500 **Guard Flags**

CF Instrumented

FID table present

Long jump target table present

0000 Code Integrity Flags

0000 Code Integrity Catalog

00000000 Code Integrity Catalog Offset

00000000 Code Integrity Reserved

0000000000000000 **Guard CF address taken IAT entry table**

0 **Guard CF address taken IAT entry count**

0000000000000000 **Guard CF long jump target table**

0 **Guard CF long jump target count**

0000000000000000 Dynamic value relocation table

Guard CF Function Table

Address

--------

0000000140001010 \_TlgEnableCallback

0000000140001070 SmpSessionComplete

0000000140001080 NtProcessStartupW

0000000140001B30 SmscpLoadSubSystemsForMuSession

0000000140001D10 SmscpExecuteInitialCommand

0000000140002FB0 SmpExecPgm

0000000140003620 SmpStartCsr

00000001400039F0 SmpApiCallback

0000000140004E90 SmpStopCsr

...

The CFG-related information is marked in bold in the preceding output. We will discuss that shortly. For now, open Process Explorer, right-click the process column header, choose **Select Columns.** Then, in the Process Image tab, select the **Control Flow Guard** check box. Also select **Virtual Size** in the Process Memory tab. You should see something like this:

Image

You should see most Microsoft-provided processes were built with CFG (including Smss, Csrss, Audiodg, Notepad, and many others). The virtual size for CFG-built processes is surprisingly high. Recall that the virtual size indicates the total address space used in the process, whether that memory is committed or reserved. In contrast, the Private Bytes column shows the private committed memory and is not even remotely close to the virtual size (although the virtual size includes non-private memory as well). For 64-bit processes, the virtual size is at least 2 TB, which we will shortly be able to rationalize.

##### **The CFG bitmap**

As you saw earlier, forcing the program to iterate through a list of function calls every few instructions would not be practical. Therefore, instead of an algorithm that requires linear time O(*n*), performance requirements dictate that an O(1) algorithm be used instead—one where a constant lookup time is used, regardless of how many functions are present in the table. This constant lookup time should be as small as possible. A clear winner of such a requirement would be an array that is indexable by the target function’s address, which is an indication if this address is valid or not (such as a simple BOOL). With a 128 TB of possible addresses, though, such an array would itself have to be 128 TB \* sizeof(BOOL), which is an unacceptable storage size—bigger than the address space itself. Can we do better?

First, we can leverage the fact that compilers ought to generate x64 function code on 16-byte boundaries. This reduces the size to the required array to only 8 TB \* sizeof(BOOL). But using an entire BOOL (which is 4 bytes in the worst case or 1 byte in the best) is extremely wasteful. We only need one state, valid or invalid, which only needs to use 1 bit. This makes the calculation 8 TB / 8, or simply 1 TB. Unfortunately, however, there’s a snag. There’s no guarantee that the compiler will generate *all* functions on a 16-byte binary. Hand-crafted assembly code and certain optimizations might violate this rule. As such, we’ll have to figure out a solution. One possible option is to simply use another bit to indicate if the function begins *somewhere* on the next 15 bytes instead of on the 16-byte boundary itself. Thus, we have the following possibilities:

Image **{0, 0}** No valid function begins inside this 16-byte boundary.

Image **{1, 0}** A valid function begins exactly on this aligned 16-byte address.

Image **{1, 1}** A valid function begins somewhere inside of this 16-byte address.

Thanks to this setup, if the attacker attempts to call inside a function that was marked as 16-byte aligned by the linker, the 2-bit state will be {1, 0}, while the required bits (that is, bits 3 and 4) in the address will be {1, 1} as the address won’t be 16-byte aligned. Therefore, an attacker will only be able to call an arbitrary instruction in the first 16 bytes of the function if the linker did not generate the function aligned in the first place (bits would then be {1, 1}, as shown above). Even then, this instruction must somehow be useful to the attacker without crashing the function (typically some sort of stack pivot or gadget that ends in a ret instruction).

With this understanding in mind, we can apply the following formulas to compute the size of the CFG bitmap:

Image **32-bit application on x86 or x64** 2 GB / 16 \* 2 = 32 MB

Image **32-bit application with /LARGEADDRESSAWARE, booted in 3 GB mode on x86** 3 GB / 16 \* 2 = 48 MB

Image **64-bit application** 128 TB / 16 \* 2 = 2 TB

Image **32-bit application with /LARGEADDRESSAWARE, on x64** 4 GB / 16 \* 2 = 64 MB, plus the size of the 64-bit bitmap, which is needed to protect 64-bit Ntdll.dll and WoW64 components, so 2 TB + 64MB

Allocating and filling out 2 TB of bits on every single process execution is still a tough performance overhead to swallow. Even though we have fixed the execution cost of the indirect call itself, process startup cannot be allowed to take so long, and 2 TB of committed memory would exhaust the commit limit instantly. Therefore, two memory-saving and performance-helping tricks are used.

First, the memory manager will only reserve the bitmap, basing itself on the assumption that the CFG validation function will treat an exception during CFG bitmap access as an indication that the bit state is {0,0}. As such, as long as the region contains 4 KB of bit states that are all {0, 0}, it can be left as reserved, and only pages with at least one bit set {1, X} need to be committed.

Next, as described in the ASLR section of [Chapter 5](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#ch05), “[Memory management](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#ch05),” the system performs the randomization/relocation of libraries typically only once at boot, as a performance-saving measure to avoid repeated relocations. As such, after a library that supports ASLR has been loaded once at a given address, it will always be loaded at that same address. This also therefore means that once the relevant bitmap states have been calculated for the functions in that library, they will be identical in all other processes that also load the same binary. As such, the memory manager treats the CFG bitmap as a region of pagefile-backed shareable memory, and the physical pages that correspond to the shared bits only exist in RAM once.

This reduces the cost of the committed pages in RAM and means that only the bits corresponding to private memory need to be calculated. In regular applications, private memory is not executable except in the copy-on-write case where someone has patched a library (but this will not happen at image load), so the cost of loading an application, if it shares the same libraries as other previously launched applications, is almost nil. The next experiment demonstrates this.

EXPERIMENT: Control Flow Guard bitmap

Open the VMMap tool and select a Notepad process. You should see a large reserved block in the Sharable section like so:

Image

You can sort the lower pane by size and quickly locate the large chunk used for the CFGBitmap, as shown. Additionally, if you attach to the process and use the !address command on the process, you will see WinDBG identifying the CFG bitmap for you:

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p746pro01a)

+ 7df5'ff530000 7df6'0118a000 0'01c5a000 MEM\_MAPPED MEM\_RESERVE

Other [CFG Bitmap]

7df6'0118a000 7df6'011fb000 0'00071000 MEM\_MAPPED MEM\_COMMIT PAGE\_NOACCESS

Other [CFG Bitmap]

7df6'011fb000 7ff5'df530000 1ff'de335000 MEM\_MAPPED MEM\_RESERVE

Other [CFG Bitmap]

7ff5'df530000 7ff5'df532000 0'00002000 MEM\_MAPPED MEM\_COMMIT PAGE\_READONLY

Other [CFG Bitmap]

Note how large regions are marked as MEM\_RESERVE, in between regions that are MEM\_COMMIT, representing that at least one valid bit state {1, X} is set. Also, all (or almost all) the regions will be MEM\_MAPPED, since they belong to the shared bitmap.

##### **CFG bitmap construction**

Upon system initialization, the MiInitializeCfg function is called to initialize support for CFG. The function creates one or two section objects (MmCreateSection) as reserved memory with size appropriate for the platform, as shown earlier. For 32-bit platforms, one bitmap is enough. For x64 platforms, two bitmaps are required—one for 64-bit processes and the other for Wow64 processes (32-bit applications). The section objects’ pointers are stored in a substructure within the MiState global variable.

After a process is created, the appropriate section is securely mapped into the process’s address space. *Securely* here means that the section cannot be unmapped by code running within the process or have its protection changed. (Otherwise, malicious code could just unmap the memory, reallocate, and fill everything with 1 bits, effectively disabling CFG, or simply modify any bits by marking the region read/write.)

The user mode CFG bitmap(s) are populated in the following scenarios:

Image During image mapping, images that have been dynamically relocated due to ASLR (see [Chapter 5](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#ch05), for more on ASLR) will have their indirect call target metadata extracted. If an image does not have indirect call target metadata, meaning it was not compiled with CFG, it is assumed that every address within the image can be called indirectly. As explained, because dynamically relocated images are expected to load at the same address in every process, their metadata is used to populate the shared section that is used for the CFG bitmap.

Image During image mapping, special care is needed for non-dynamically relocated images and images not being mapped at their preferred base. For these image mappings, the relevant pages of the CFG bitmap are made private and are populated using the CFG metadata from the image. For images whose CFG bits are present in the shared CFG bitmap, a check is made to ensure that all the relevant CFG bitmap pages are still shared. If this is not the case, the bits of the private CFG bitmap pages are populated using the CFG metadata from the image.

Image When virtual memory is allocated or re-protected as executable, the relevant pages of the CFG bitmap are made private and initialized to all 1s by default. This is needed for cases such as just-in-time (JIT) compilation, where code is generated on the fly and then executed (for example, .NET or Java).

##### **Strengthening CFG protection**

Although CFG does an adequate job to prevent types of exploits that leverage indirect calls or jumps, it could be bypassed through the following ways:

Image If the process can be tricked or an existing JIT engine abused to allocate executable memory, all the corresponding bits will be set to {1, 1}, meaning that all memory is considered a valid call target.

Image For 32-bit applications, if the expected call target is \_\_stdcall (standard calling convention), but an attacker is able to change the indirect call target to \_\_cdecl (C calling convention), the stack will become corrupt, as the C call function will not perform cleanup of the caller’s arguments, unlike a standard call function. Because CFG cannot differentiate between the different calling conventions, this results in a corrupt stack, potentially with an attacker-controlled return address, bypassing the CFG mitigation.

Image Similarly, compiler-generated setjmp/longjmp targets behave differently from true indirect calls. CFG cannot differentiate between the two.

Image Certain indirect calls are harder to protect, such as the Import Address Table (IAT) or Delay-Load Address Table, which is typically in a read-only section of the executable.

Image Exported functions may not be desirable indirect function calls.

Windows 10 introduces advancements to CFG that address all these issues. The first is to introduce a new flag to the VirtualAlloc function called PAGE\_TARGETS\_INVALID and one to VirtualProtect called PAGE\_TARGETS\_NO\_UPDATE. With these flags set, JIT engines that allocate executable memory will not see all their allocations’ bits set to the {1, 1} state. Instead, they must manually call the SetProcess-ValidCallTargets function (which calls the native NtSetInformationVirtualMemory function), which will allow them to specify the actual function start addresses of their JITed code. Additionally, this function is marked as a suppressed call with DECLSPEC\_GUARD\_SUPPRESS, making sure that attackers cannot use an indirect CALL or JMP to redirect into it, even at its function start. (Because it’s an inherently dangerous function, calling it with a controlled stack or registers could result in the bypassing of CFG.)

Next, improved CFG changes the default flow you saw in the beginning of this section with a more refined flow. In this flow, the loader does not implement a simple “verify target, return” function, but rather a “verify target, call target, check stack, return” function, which is used in a subset of places on 32-bit applications (and/or running under WoW64). This improved execution flow is shown in [Figure 7-32](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig32).

Image

**FIGURE 7-32** Improved CFG.

Next, improved CFG adds additional tables inside of the executable, such as the Address Taken IAT table and the Long Jump Address table. When longjmp and IAT CFG protection are enabled in the compiler, these tables are used to store destination addresses for these specific types of indirect calls, and the relevant functions are *not* placed in the regular function table, therefore not figuring in the bitmap. This means that if code is attempting to indirect jump/call to one of these functions, it will be treated as an illegal transition. Instead, the C Runtime and linker will validate the targets of, say, the longjmp function, by manually checking this table. Although it’s more inefficient than a bitmap, there should be little to no functions in these tables, making the cost bearable.

Finally, improved CFG implements a feature called *export suppression*, which must be supported by the compiler and enabled by process-mitigation policy. (See the section “[Process-mitigation policies](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07lev2sec26)” for more on process level mitigations.) With this feature enabled, a new bit state is implemented (recall that bulleted list had {0, 1} as an undefined state). This state indicates that the function is valid but export-suppressed, and it will be treated differently by the loader.

You can determine which features are present in a given binary by looking at the guard flags in the Image Load Configuration Directory, which the DumpBin application used earlier can decode. For reference, they are listed in [Table 7-21](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07tab21).

Image

**TABLE 7-21** Control Flow Guard flags

##### **Loader interaction with CFG**

Although it is the memory manager that builds the CFG bitmap, the user-mode loader (see [Chapter 3](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#ch03) for more information) serves two purposes. The first is to dynamically enable CFG support only if the feature is enabled (for example, the caller may have requested no CFG for the child process, or the process itself might not have CFG support). This is done by the LdrpCfgProcessLoadConfig loader function, which is called to initialize CFG for each loaded module. If the module DllCharacteristics flags in the optional header of the PE does not have the CFG flag set (IMAGE\_DLLCHARACTERISTICS\_GUARD\_CF), the GuardFlags member of IMAGE\_LOAD\_CONFIG\_DIRECTORY structure does not have the IMAGE\_GUARD\_CF\_INSTRUMENTED flag set, or the kernel has forcibly turned off CFG for this module, then there is nothing to do.

Second, if the module is indeed using CFG, LdrpCfgProcessLoadConfig gets the indirect checking function pointer retrieved from the image (the GuardCFCheckFunctionPointer member of IMAGE\_LOAD\_CONFIG\_DIRECTORY structure) and sets it to either LdrpValidateUserCallTarget or LdrpValidateUserCallTargetES in Ntdll, depending on whether export suppression is enabled. Additionally, the function first makes sure the indirect pointer has not been somehow modified to point outside the module itself.

Furthermore, if improved CFG was used to compile this binary, a second indirect routine is available, called the *dispatch CFG routine*. It is used to implement the enhanced execution flow described earlier. If the image includes such a function pointer (in the GuardCFDispatchFunctionPointer member of the abovementioned structure), it is initialized to LdrpDispatchUserCallTarget, or LdrpDispatchUserCallTargetES if export suppression is enabled.

Image Note

In some cases, the kernel itself can emulate or perform indirect jumps or calls on behalf of user mode. In situations where this is a possibility, the kernel implements its own MmValidateUserCallTarget routine, which performs the same work as LdrpValidate-UserCallTarget.

The code generated by the compiler when CFG is enabled issues an indirect call that lands in the LdrpValidateCallTarget(ES) or LdrpDispatchUserCallTarget(ES) functions in Ntdll. This function uses the target branch address and checks the bit state value for the function:

Image If the bit state is {0, 0}, the dispatch is potentially invalid.

Image If the bit state is {1, 0}, and the address is 16-byte aligned, the dispatch is valid. Otherwise, it is potentially invalid.

Image If the bit state is {1, 1}, and the address is not 16-byte aligned, the dispatch is valid. Otherwise, it is potentially invalid.

Image If the bit state is {0, 1}, the dispatch is potentially invalid.

If the dispatch is potentially invalid, the RtlpHandleInvalidUserCallTarget function will execute to determine the appropriate action. First, it checks if suppressed calls are allowed in the process, which is an unusual application-compatibility option that might be set if Application Verifier is enabled, or through the registry. If so, it will check if the address is suppressed, which is why it was not inserted into the bitmap (recall that a special flag in the guard function table entry indicates this). If this is the case, the call is allowed through. If the function is not valid at all (meaning it’s not in the table), then the dispatch is aborted and the process terminated.

Second, a check is made to see if export suppression is enabled. If it is, the target address is checked against the list of export-suppressed addresses, which is once again indicated with another flag that is added in the guard function table entry. If this is the case, the loader validates that the target address is a forwarder reference to the export table of another DLL, which is the only allowed case of an indirect call toward an image with suppressed exports. This is done by a complex check that makes sure the target address is in a different image, that its image load directory has enabled export suppression, and that this address is in the import directory of that image. If these checks match, the kernel is called through the NtSetInformationVirtualMemory call described earlier, to change the bit state to {1, 0}. If any of these checks fail, or export suppression is not enabled, then the process is terminated.

For 32-bit applications, an additional check is performed if DEP is enabled for the process. (See [Chapter 5](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#ch05) for more on DEP.) Otherwise, because there are no execution guarantees to begin with, the incorrect call is allowed, as it may be an older application calling into the heap or stack for legitimate reasons.

Finally, because large sets of {0, 0} bit states are not committed to save space, if checking the CFG bitmap lands on a reserved page, an access violation exception occurs. On x86, where exception handling setup is expensive, instead of being handled as part of the verification code, it is left to propagate normally. (See Chapter 8 in Part 2 for more on exception dispatching.) The user-mode dispatcher handler, KiUserExceptionDispatcher, has specific checks for recognizing CFG bitmap access violation exceptions within the validation function and will automatically resume execution if the exception code was STATUS\_IN\_PAGE\_ERROR. This simplifies the code in LdrpValidateUserCallTarget(ES) and LdrpDispatchUserCallTarget(ES), which don’t have to include exception handling code. On x64, where exception handlers are simply registered in tables, the LdrpICallHandler handler runs instead, with the same logic as above.

##### **Kernel CFG**

Although drivers compiled with Visual Studio and /guard:cf also ended up with the same binary properties as user-mode images, the first versions of Windows 10 did not do anything with this data. Unlike the user-mode CFG bitmap, which is protected by a higher, more trusted entity (the kernel), there is nothing that can truly “protect” the kernel CFG bitmap if one were to be created. A malicious exploit could simply edit the PTE that corresponded to the page containing the desired bits to modify, mark it as read/write, and proceed with the indirect call or jump. Therefore, the overhead of setting up such a trivially bypassable mitigation was simply not worth it.

With a greater number of users enabling VBS features, once again, the higher security boundary that VTL 1 provides can be leveraged. The SLAT page table entries come to the rescue by providing a second boundary against PTE page protection changes. While the bitmap is readable to VTL 0 because the SLAT entries are marked as read only, if a kernel attacker attempts to change the PTEs to mark them read/write, they cannot do the same to the SLAT entries. As such, this will be detected as an invalid KCFG bitmap access which HyperGuard can act on (for telemetry reasons alone—since the bits can’t be changed anyway).

KCFG is implemented almost identically to regular CFG, except that export suppression is not enabled, nor is longjmp support, nor is the ability to dynamically request additional bits for JIT purposes. Kernel drivers should not be doing any of these things. Instead, the bits are set in the bitmap based on the “address taken IAT table” entries, if any are set; by the usual function entries in the guard table each time a driver image is loaded; and for the HAL and kernel during boot by MiInitializeKernelCfg. If the hypervisor is not enabled, and SLAT support is not present, then none of this will be initialized, and Kernel CFG will be kept disabled.

Just like in the user-mode case, a dynamic pointer in the load configuration data directory is updated, which in the enabled case will point to \_\_guard\_check\_icall for the check function and \_\_guard\_dispatch\_icall for the dispatch function in enhanced CFG mode. Additionally, a variable named guard\_icall\_bitmap will hold the virtual address of the bitmap.

One last detail on Kernel CFG is that unfortunately, dynamic Driver Verifier settings will not be configurable (for more information on Driver Verifier, see [Chapter 6](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#ch06), “[I/O system](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#ch06)”), as this would require adding dynamic kernel hooks and redirecting execution to functions that may not be in the bitmap. In this case, STATUS\_VRF\_CFG\_ENABLED (0xC000049F) will be returned, and a reboot is required (at which time the bitmap can be built with the Verifier Driver hooks in place).

#### **Security assertions**

Earlier, we described how Control Flow Guard will terminate the process. We also explained how certain other mitigations or security features will raise an exception to kill the process. It is important to be accurate with what *exactly* happens during these security violations because both these descriptions hide important details about the mechanism.

In fact, when a security-related breach occurs, such as when CFG detects an incorrect indirect call or jump, terminating the process through the standard TerminateProcess mechanism would not be an adequate path. There would be no crash generated, and no telemetry sent to Microsoft. These are both important tools for the administrator to understand that a potential exploit has executed or that an application compatibility issue exists, as well as for Microsoft to track zero-day exploitation in the wild. On the flip side, while raising an exception would achieve the desired result, exceptions are callbacks, which can be:

Image Potentially hooked by attackers if /SAFESEH and SEHOP mitigations are not enabled, causing the security check to be the one that gives control to an attacker in the first place—or an attacker can simply “swallow” the exception.

Image Potentially hooked by legitimate parts of the software through an unhandled exception filter or vectored exception handler, both of which might accidentally swallow the exception.

Image Same as above, but intercepted by a third-party product that has injected its own library into the process. Common to many security tools, this can also lead in the exception not being correctly delivered to Windows Error Reporting (WER).

Image A process might have an application recovery callback registered with WER. This might then display a less clear UI to the user, and might restart the process in its current exploited state, leading anywhere from a recursive crash/start loop to the exception being swallowed whole.

Image Likely in a C++-based product, caught by an outer exception handler as if “thrown” by the program itself, which, once again, might swallow the exception or continue execution in an unsafe manner.

Solving these issues requires a mechanism that can raise an exception that cannot be intercepted by any of the process’s components outside of the WER service, which must itself be guaranteed to receive the exception. This is where security assertions come into play.

##### **Compiler and OS support**

When Microsoft libraries, programs, or kernel components encounter unusual security situations, or when mitigations recognize dangerous violations of security state, they now use a special compiler intrinsic supported by Visual Studio, called \_\_fastfail, which takes one parameter as input. Alternatively, they can call a runtime library (Rtl) function in Ntdll called RtlFailFast2, which itself contains a \_\_fastfail intrinsic. In some cases, the WDK or SDK contain inline functions that call this intrinsic, such as when using the LIST\_ENTRY functions InsertTailList and RemoveEntryList. In other situations, it is the Universal CRT (uCRT) itself that has this intrinsic in its functions. In yet others, APIs will do certain checks when called by applications and may use this intrinsic as well.

Regardless of the situation, when the compiler sees this intrinsic, it generates assembly code that takes the input parameter, moves it into the RCX (x64) or ECX (x86) register, and then issues a software interrupt with the number 0x29. (For more information on interrupts, see Chapter 8 in Part 2.)

In Windows 8 and later, this software interrupt is registered in the Interrupt Dispatch Table (IDT) with the handler KiRaiseSecurityCheckFailure, which you can verify on your own by using the !idt 29 command in the debugger. This will result (for compatibility reasons) in KiFastFailDispatch being called with the STATUS\_STACK\_BUFFER\_OVERRUN status code (0xC0000409). This will then do regular exception dispatching through KiDispatchException, but treat this as a second-chance exception, which means that the debugger and process won’t be notified.

This condition will be specifically recognized and an error message will be sent to the WER error ALPC port as usual. WER will claim the exception as non-continuable, which will then cause the kernel to terminate the process with the usual ZwTerminateProcess system call. This, therefore, guarantees that once the interrupt is used, no return to user mode will ever be performed within this process again, that WER will be notified, and that the process will be terminated (additionally, the error code will be the exception code). When the exception record is generated, the first exception argument will be the input parameter to \_\_fastfail.

Kernel-mode code can also raise exceptions, but in this case KiBugCheckDispatch will be called instead, which will result in a special kernel mode crash (bugcheck) with code 0x139 (KERNEL\_SECURITY\_CHECK\_FAILURE), where the first argument will be the input parameter to \_\_fastfail.

##### **Fast fail/security assertion codes**

Because the \_\_fastfail intrinsic contains an input argument that is bubbled up to the exception record or crash screen, it allows the failing check to identify what part of the system or process is not working correctly or has encountered a security violation. [Table 7-22](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07tab22) shows the various failure conditions and their meaning or significance.

Image

Image

Image

Image

**TABLE 7-22** \_\_fastfail failure codes

### **APPLICATION IDENTIFICATION**

Historically, security decisions in Windows have been based on a user’s identity (in the form of the user’s SID and group membership), but a growing number of security components (AppLocker, firewall, antivirus, anti-malware, Rights Management Services, and others) need to make security decisions based on what code is to be run. In the past, each of these security components used their own proprietary method for identifying applications, which led to inconsistent and overly complicated policy authoring. The purpose of Application Identification (AppID) is to bring consistency to how the security components recognize applications by providing a single set of APIs and data structures.

Image Note

This is not the same as the AppID used by DCOM/COM+ applications, where a GUID represents a process that is shared by multiple CLSIDs, nor is it related to UWP application ID.

Just as a user is identified when she logs in, an application is identified just before it is started by generating the main program’s AppID. An AppID can be generated from any of the following attributes of the application:

Image **Fields** Fields within a code-signing certificate embedded within the file allow for different combinations of publisher name, product name, file name, and version. APPID://FQBN is a fully qualified binary name, and it is a string in the following form: {Publisher\Product\Filename,Version}. Publisher is the Subject field of the x.509 certificate used to sign the code, using the following fields:

• **O** Organization

• **L** Locality

• **S** State or province

• **C** Country

Image **File hash** there are several methods that can be used for hashing. The default is APPID://SHA256HASH. However, for backward compatibility with SRP and most x.509 certificates, SHA-1 (APPID://SHA1HASH) is still supported. APPID://SHA256HASH specifies the SHA-256 hash of the file.

Image **The partial or complete path to the file** APPID://Path specifies a path with optional wildcard characters (\*).

Image Note

An AppID does not serve as a means for certifying the quality or security of an application. An AppID is simply a way of identifying an application so that administrators can reference the application in security policy decisions.

The AppID is stored in the process access token, allowing any security component to make authorization decisions based on a single consistent identification. AppLocker uses conditional ACEs (described earlier) for specifying whether a particular program is allowed to be run by the user.

When an AppID is created for a signed file, the certificate from the file is cached and verified to a trusted root certificate. The certificate path is reverified daily to ensure the certificate path remains valid. Certificate caching and verification are recorded in the system event log at Application and Services Logs\Microsoft\Windows\AppID\Operational.

### **APPLOCKER**

Windows 8.1 and Windows 10 (Enterprise editions) and Windows Server 2012/R2/2016 support a feature known as *AppLocker*, which allows an administrator to lock down a system to prevent unauthorized programs from being run. Windows XP introduced Software Restriction Policies (SRP), which was the first step toward this capability, but SRP was difficult to manage, and it couldn’t be applied to specific users or groups. (All users were affected by SRP rules.) AppLocker is a replacement for SRP, and yet coexists alongside SRP, with AppLocker’s rules being stored separately from SRP’s rules. If both AppLocker and SRP rules are in the same Group Policy object (GPO), only the AppLocker rules will be applied.

Another feature that makes AppLocker superior to SRP is AppLocker’s auditing mode, which allows an administrator to create an AppLocker policy and examine the results (stored in the system event log) to determine whether the policy will perform as expected—without actually performing the restrictions. AppLocker auditing mode can be used to monitor which applications are being used by one or more users on a system.

AppLocker allows an administrator to restrict the following types of files from being run:

Image Executable images (EXE and COM)

Image Dynamic-link libraries (DLL and OCX)

Image Microsoft Software Installer (MSI and MSP) for both install and uninstall

Image Scripts

Image Windows PowerShell (PS1)

Image Batch (BAT and CMD)

Image VisualBasic Script (VBS)

Image Java Script (JS)

AppLocker provides a simple GUI rule-based mechanism, which is very similar to network firewall rules, for determining which applications or scripts are allowed to be run by specific users and groups, using conditional ACEs and AppID attributes. There are two types of rules in AppLocker:

Image Allow the specified files to run, denying everything else.

Image Deny the specified files from being run, allowing everything else. Deny rules take precedence over allow rules.

Each rule can also have a list of exceptions to exclude files from the rule. Using an exception, you could create a rule to, for example, allow everything in the C:\Windows or C:\Program Files directories to be run except RegEdit.exe.

AppLocker rules can be associated with a specific user or group. This allows an administrator to support compliance requirements by validating and enforcing which users can run specific applications. For example, you can create a rule to allow users in the Finance security group to run the finance line-of-business applications. This blocks everyone who is not in the Finance security group from running finance applications (including administrators) but still provides access for those who have a business need to run the applications. Another useful rule would be to prevent users in the Receptionists group from installing or running unapproved software.

AppLocker rules depend upon conditional ACEs and attributes defined by AppID. Rules can be created using the following criteria:

Image **Fields within a code-signing certificate embedded within the file, allowing for different combinations of publisher name, product name, file name, and version** For example, a rule could be created to allow all versions greater than 9.0 of Contoso Reader to run or allow anyone in the Graphics group to run the installer or application from Contoso for GraphicsShop as long as the version is 14.\*. For example, the following SDDL string denies execute access to any signed programs published by Contoso for the RestrictedUser user account (identified by the user’s SID):

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p759pro01a)

D:(XD;;FX;;;S-1-5-21-3392373855-1129761602-2459801163-1028;((Exists APPID://FQBN)

&& ((APPID://FQBN) >= ({"O=CONTOSO, INCORPORATED, L=REDMOND,

S=CWASHINGTON, C=US\\*\\*",0}))))

Image **Directory path, allowing only files within a particular directory tree to run** This can also be used to identify specific files. For example, the following SDDL string denies execute access to the programs in the directory C:\Tools for the RestrictedUser user account (identified by the user’s SID):

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p759pro02a)

D:(XD;;FX;;;S-1-5-21-3392373855-1129761602-2459801163-1028;(APPID://PATH

Contains "%OSDRIVE%\TOOLS\\*"))

Image **File hash** Using a hash will also detect if a file has been modified and prevent it from running. This can also be a weakness if files are changed frequently because the hash rule will need to be updated frequently. File hashes are often used for scripts because few scripts are signed. For example, this SDDL string denies execute access to programs with the specified hash values for the RestrictedUser user account (identified by the user’s SID):

[**Click here to view code image**](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07_img.html#p759pro03a)

D:(XD;;FX;;;S-1-5-21-3392373855-1129761602-2459801163-1028;(APPID://SHA256HASH

Any\_of {#7a334d2b99d48448eedd308dfca63b8a3b7b44044496ee2f8e236f5997f1b647,

#2a782f76cb94ece307dc52c338f02edbbfdca83906674e35c682724a8a92a76b}))

AppLocker rules can be defined on the local machine using the Security Policy MMC snap-in (secpol.msc, see [Figure 7-33](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig33)) or a Windows PowerShell script, or they can be pushed to machines within a domain using Group Policy. AppLocker rules are stored in multiple locations within the registry:

Image **HKLM\Software\Policies\Microsoft\Windows\SrpV2** This key is also mirrored to HKLM\SOFTWARE\Wow6432Node\Policies\Microsoft\Windows\SrpV2. The rules are stored in XML format.

Image **HKLM\SYSTEM\CurrentControlSet\Control\Srp\Gp\Exe** The rules are stored as SDDL and a binary ACE.

Image **HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Group Policy Objects\{GUID}Machine\Software\Policies\Microsoft\Windows\SrpV2** AppLocker policy pushed down from a domain as part of a GPO are stored here in XML format.

Image

**FIGURE 7-33** AppLocker configuration page in Local Security Policy.

Certificates for files that have been run are cached in the registry under the key HKLM\SYSTEM\CurrentControlSet\Control\AppID\CertStore. AppLocker also builds a certificate chain (stored in HKLM\SYSTEM\CurrentControlSet\Control\AppID\CertChainStore) from the certificate found in a file back to a trusted root certificate.

There are also AppLocker-specific PowerShell commands (cmdlets) to enable deployment and testing via scripting. After using the Import-Module AppLocker to get AppLocker cmdlets into PowerShell, several cmdlets are available. These include Get-AppLockerFileInformation, Get-AppLockerPolicy, New-AppLockerPolicy, Set-AppLockerPolicy, and Test-AppLockerPolicy.

The AppID and SRP services coexist in the same binary (AppIdSvc.dll), which runs within an SvcHost process. The service requests a registry change notification to monitor any changes under that key, which is written by either a GPO or the AppLocker UI in the Local Security Policy MMC snap-in. When a change is detected, the AppID service triggers a user-mode task (AppIdPolicyConverter.exe), which reads the new rules (described with XML) and translates them into binary format ACEs and SDDL strings, which are understandable by both the user-mode and kernel-mode AppID and AppLocker components. The task stores the translated rules under HKLM\SYSTEM\CurrentControlSet\Control\Srp\Gp. This key is writable only by System and Administrators, and it is marked read-only for authenticated users. Both user-mode and kernel-mode AppID components read the translated rules from the registry directly. The service also monitors the local machine trusted root certificate store, and it invokes a user-mode task (AppIdCertStoreCheck.exe) to reverify the certificates at least once per day and whenever there is a change to the certificate store. The AppID kernel-mode driver (%SystemRoot%\System32\drivers\AppId.sys) is notified about rule changes by the AppID service through an APPID\_POLICY\_CHANGED DeviceIoControl request.

An administrator can track which applications are being allowed or denied by looking at the system event log using Event Viewer (once AppLocker has been configured and the service started). See [Figure 7-34](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig34).

Image

**FIGURE 7-34** Event Viewer showing AppLocker allowing and denying access to various applications. Event ID 8004 is denied; 8002 is allowed.

The implementations of AppID, AppLocker, and SRP are somewhat blurred and violate strict layering, with various logical components coexisting within the same executables, and the naming is not as consistent as one would like.

The AppID service runs as LocalService so that it has access to the Trusted Root Certificate Store on the system. This also enables it to perform certificate verification. The AppID service is responsible for the following:

Image Verification of publisher certificates

Image Adding new certificates to the cache

Image Detecting AppLocker rule updates and notifying the AppID driver

The AppID driver performs the majority of the AppLocker functionality and relies on communication (via DeviceIoControl requests) from the AppID service, so its device object is protected by an ACL, granting access only to the NT SERVICE\AppIDSvc, LOCAL SERVICE and BUILTIN\Administrators groups. Thus, the driver cannot be spoofed by malware.

When the AppID driver is first loaded, it requests a process-creation callback by calling PsSetCreateProcessNotifyRoutineEx. When the notification routine is called, it is passed a PPS\_CREATE\_NOTIFY\_INFO structure (describing the process being created). It then gathers the AppID attributes that identify the executable image and writes them to the process’s access token. Then it calls the undocumented routine SeSrpAccessCheck, which examines the process token and the conditional ACE AppLocker rules, and determines whether the process should be allowed to run. If the process should not be allowed to run, the driver writes STATUS\_ACCESS\_DISABLED\_BY\_POLICY\_OTHER to the Status field of the PPS\_CREATE\_NOTIFY\_INFO structure, which causes the process creation to be canceled (and sets the process’s final completion status).

To perform DLL restriction, the image loader sends a DeviceIoControl request to the AppID driver whenever it loads a DLL into a process. The driver then checks the DLL’s identity against the AppLocker conditional ACEs, just like it would for an executable.

Image Note

Performing these checks for every DLL load is time-consuming and might be noticeable to end users. For this reason, DLL rules are normally disabled, and they must be specifically enabled via the Advanced tab in the AppLocker properties page in the Local Security Policy snap-in.

The scripting engines and the MSI installer have been modified to call the user-mode SRP APIs whenever they open a file, to check whether a file is allowed to be opened. The user-mode SRP APIs call the AuthZ APIs to perform the conditional ACE access check.

### **SOFTWARE RESTRICTION POLICIES**

Windows contains a user-mode mechanism called *Software Restriction Policies* (*SRP*) that enables administrators to control what images and scripts execute on their systems. The Software Restriction Policies node of the Local Security Policy editor, shown in [Figure 7-35](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07fig35), serves as the management interface for a machine’s code execution policies, although per-user policies are also possible using domain group policies.

Image

**FIGURE 7-35** Software Restriction Policy configuration.

Several global policy settings appear beneath the Software Restriction Policies node:

Image **Enforcement** This policy configures whether restriction policies apply to libraries, such as DLLs, and whether policies apply to users only or to administrators as well.

Image **Designated File Types** This policy records the extensions for files that are considered executable code.

Image **Trusted Publishers** This policy controls who can select which certificate publishers are trusted.

When configuring a policy for a particular script or image, an administrator can direct the system to recognize it using its path, its hash, its Internet zone (as defined by Internet Explorer), or its cryptographic certificate, and can specify whether it is associated with the Disallowed or Unrestricted security policy.

Enforcement of SRPs takes place within various components where files are treated as containing executable code. Some of these components are listed here:

Image The user-mode Windows CreateProcess function in Kernel32.dll enforces it for executable images.

Image The DLL loading code in Ntdll enforces it for DLLs.

Image The Windows command prompt (Cmd.exe) enforces it for batch file execution.

Image Windows Scripting Host components that start scripts—Cscript.exe (for command-line scripts), Wscript.exe (for UI scripts), and Scrobj.dll (for script objects)—enforce it for script execution.

Image The PowerShell host (PowerShell.exe) enforces it for PowerShell script execution.

Each of these components determines whether the restriction policies are enabled by reading the TransparentEnabled registry value in the HKLM\Software\Policies\Microsoft \Windows\Safer\CodeIdentifiers key, which if set to 1 indicates that policies are in effect. Then it determines whether the code it’s about to execute matches one of the rules specified in a subkey of the CodeIdentifiers key and, if so, whether the execution should be allowed. If there is no match, the default policy, as specified in the DefaultLevel value of the CodeIdentifiers key, determines whether the execution is allowed.

Software Restriction Policies are a powerful tool for preventing the unauthorized access of code and scripts, but only if properly applied. Unless the default policy is set to disallow execution, a user can make minor changes to an image that’s been marked as disallowed so that he can bypass the rule and execute it. For example, a user can change an innocuous byte of a process image so that a hash rule fails to recognize it, or copy a file to a different location to avoid a path-based rule.

EXPERIMENT: Watching Software Restriction Policy enforcement

You can indirectly see SRPs being enforced by watching accesses to the registry when you attempt to execute an image that you’ve disallowed.

**1.** Run secpol.msc to open the Local Security Policy editor and navigate to the Software Restriction Policies node.

**2.** Choose **Create New Policies** from the context menu if no policies are defined.

**3.** Create a path-based disallow restriction policy (under the Additional Rules node) for %SystemRoot%\System32\Notepad.exe.

**4.** Run Process Monitor and set an include a Path filter for Safer.

**5.** Open a command prompt and run Notepad from the prompt.

Your attempt to run Notepad should result in a message telling you that you cannot execute the specified program, and Process Monitor should show the command prompt (cmd.exe) querying the local machine restriction policies.

### **KERNEL PATCH PROTECTION**

Some device drivers modify the behavior of Windows in unsupported ways. For example, they patch the system call table to intercept system calls or patch the kernel image in memory to add functionality to specific internal functions. Such modifications are inherently dangerous and can reduce system stability and security. Additionally, it is also possible for such modifications to be made with malicious intent, either by rogue drivers or through exploits due to vulnerabilities in Windows drivers.

Without the presence of a more privileged entity than the kernel itself, detecting and protecting against kernel-based exploits or drivers from within the kernel itself is a tricky game. Because both the detection/protection mechanism and the unwanted behavior operate in ring 0, it is not possible to define a security boundary in the true sense of the word, as the unwanted behavior could itself be used to disable, patch, or fool the detection/prevention mechanism. That being said, even in such conditions, a mechanism to react to such unwanted operations can still be useful in the following ways:

Image By crashing the machine with a clearly identifiable kernel-mode crash dump, both users and administrators can easily see that an unwanted behavior has been operating inside of their kernel and they can take action. Additionally, it means that legitimate software vendors will not want to take the risk of crashing their customers’ systems and will find supported ways of extending kernel functionality (such as by using the filter manager for file system filters or other callback-based mechanisms).

Image Obfuscation (which is not a security boundary) can make it costly—either in time or in complexity—for the unwanted behavior to disable the detection mechanism. This added cost means that the unwanted behavior is more clearly identified as potentially malicious, and that its complexity results in additional costs to a potential attacker. By shifting the obfuscation techniques, it means that legitimate vendors will be better off taking the time to move away from their legacy extension mechanisms and implement supported techniques instead, without the risk of looking like malware.

Image Randomization and non-documentation of which specific checks the detection/prevention mechanism makes to monitor kernel integrity, and non-determinism of when such checks are executed, cripple the ability of attackers to ensure their exploits are reliable. It forces them to account for every possible non-deterministic variable and state transition that the mechanism has through static analysis, which obfuscation makes nearly impossible within the timeframe required before another obfuscation shift or feature change is implemented in the mechanism.

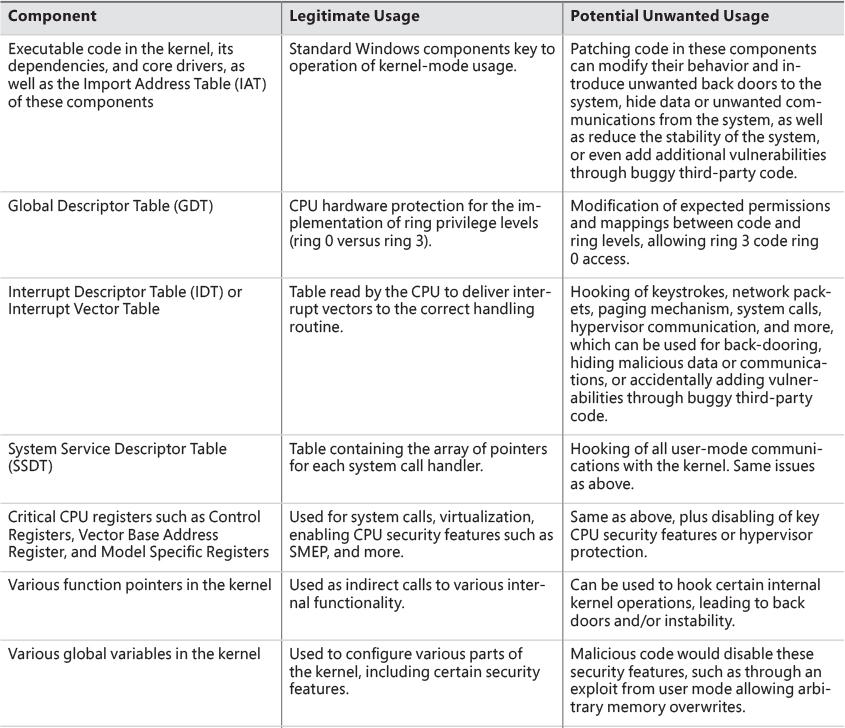
Image Because kernel mode crash dumps are automatically submitted to Microsoft, it allows the company to receive telemetry of in-the-wild unwanted code, and to either identify software vendors whose code is unsupported and is crashing systems, or to track the progress of malicious drivers in the wild, or even zero-day kernel-mode exploitations, and fix bugs that may not have been reported, but are actively exploited.

#### **PatchGuard**

Shortly after the release of 64-bit Windows for x64 and before a rich third-party ecosystem had developed, Microsoft saw an opportunity to preserve the stability of 64-bit Windows, and to add telemetry and exploit-crippling patch detection to the system, through a technology called *Kernel Patch Protection* (*KPP*), also referred to as *PatchGuard*. When Windows Mobile was released, which operates on a 32-bit ARM processor core, the feature was ported to such systems, too, and it will be present in 64-bit ARM (AArch64) systems as well. Due to the existence of too many legacy 32-bit drivers that still use unsupported and dangerous hooking techniques, however, this mechanism is not enabled on such systems, even on Windows 10 operating systems. Fortunately, usage of 32-bit systems is almost coming to an end, and server versions of Windows no longer support this architecture at all.

Although both *Guard* and *Protection* imply that this mechanism will protect the system, it is important to realize that the only guard/protection offered is the crashing of the machine, which prevents further execution of the unwanted code. The mechanism does not *prevent* the attack in the first place, nor mitigate against it, nor undo it. Think of KPP as an Internet-connected video security system, or CCTV, with a loud alarm (the crash) inside the vault (the kernel), not as an impenetrable lock on the vault.

KPP has a variety of checks that it makes on protected systems, and documenting them all would both be impractical (due to the difficulty of static analysis) and valuable to potential attackers (reducing their research time). However, Microsoft does document certain checks, which we generalize in [Table 7-23](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07tab23). When, where, and how KPP makes these checks, and which specific functions or data structures are affected, is outside of the scope of this analysis.



Image

Image

**TABLE 7-23** Generalized description of elements protected by KPP

As mentioned, when KPP detects unwanted code on the system, it crashes the system with an easily identifiable code. This corresponds to bugcheck code 0x109, which stands for CRITICAL\_STRUCTURE\_CORRUPTION, and the Windows Debugger can be used to analyze this crash dump. (See Chapter 15, “Crash dump analysis,” in Part 2 for more information.) The dump information will contain *some* information about the corrupted or scrumptiously modified part of the kernel, but any additional data must be analyzed by Microsoft’s Online Crash Analysis (OCA) and/or Windows Error Reporting (WER) teams and is not exposed to users.

For third-party developers who use techniques that KPP deters, the following supported techniques can be used:

Image **File system (mini) filters** Use these to hook all file operations, including loading image files and DLLs, that can be intercepted to purge malicious code on-the-fly or block reading of known bad executables or DLLs. (See Chapter 13, “File systems,” in Part 2 for more information on these.)

Image **Registry filter notifications** Use these to hook all registry operations. (See Chapter 9 in Part 2 for more information on these notifications.) Security software can block modification of critical parts of the registry, as well as heuristically determine malicious software by registry access patterns or known bad registry keys.

Image **Process notifications** Security software can monitor the execution and termination of all processes and threads on the system, as well as DLLs being loaded or unloaded. With the enhanced notifications added for antivirus and other security vendors, they also can block process launch. (See [Chapter 3](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#ch03) for more information on these notifications.)

Image **Object manager filtering** Security software can remove certain access rights being granted to processes and/or threads to defend their own utilities against certain operations. (These are discussed in Chapter 8 in Part 2.)

Image **NDIS Lightweight Filters (LWF) and Windows Filtering Platform (WFP) filters** Security software can intercept all socket operations (accept, listen, connect, close, and so on) and even the packets themselves. With LWF, security vendors have access to the raw Ethernet frame data that is going from the network card (NIC) to the wire.

Image **Event Tracing for Windows (ETW)** Through ETW, many types of operations that have interesting security properties can be consumed by a user-mode component, which can then react to data in near real-time. In certain cases, special secure ETW notifications are available to anti-malware-protected processes under NDA with Microsoft and participation in various security programs, which give access to a greater set of tracing data. (ETW is discussed in Chapter 8 in Part 2.)

#### **HyperGuard**

On systems that run with virtualization-based security (described earlier in this chapter in the section “[Virtualization-based security](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#ch07lev1sec3)”), it is no longer true that attackers with kernel-mode privileges are essentially running at the same security boundary as a detection/prevention mechanism. In fact, such attackers would operate at VTL 0, while a mechanism could be implemented in VTL 1. In the Anniversary Update of Windows 10 (version 1607), such a mechanism does indeed exist, which is appropriately named *HyperGuard*. HyperGuard has a few interesting properties that set it apart from PatchGuard:

Image It does not need to rely on obfuscation. The symbol files and function names that implement HyperGuard are available for anyone to see, and the code is not obfuscated. Complete static analysis is possible. This is because HyperGuard is a true security boundary.

Image It does not need to operate non-deterministically because this would provide no advantage due to the preceding property. In fact, by operating deterministically, HyperGuard can crash the system at the precise time unwanted behavior is detected. This means crash data will contain clear and actionable data for the administrator (and Microsoft’s analysis teams), such as the kernel stack, which will show the code that performed the undesirable behavior.

Image Due to the preceding property, it can detect a wider variety of attacks, because the malicious code does not have the chance to restore a value back to its correct value during a precise time window, which is an unfortunate side-effect of PatchGuard’s non-determinism.

HyperGuard is also used to extend PatchGuard’s capabilities in certain ways, and to strengthen its ability to run undetected by attackers trying to disable it. When HyperGuard detects an inconsistency, it too will crash the system, albeit with a different code: 0x18C (HYPERGUARD\_VIOLATION). As before, it might be valuable to understand, at a generic level, what kind of things HyperGuard will detect, which you can see in [Table 7-24](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#ch07tab24).

Image

**TABLE 7-24** Generalized description of elements protected by HyperGuard

On systems with VBS enabled, there is another security-related feature that is worth describing, which is implemented in the hypervisor itself: Non-Privileged Instruction Execution Prevention (NPIEP). This mitigation targets specific x64 instructions that can be used to leak the kernel-mode addresses of the GDT, IDT, and LDT, which are SGDT, SIDT, and SLDT. With NPIEP, these instructions are still allowed to execute (due to compatibility concerns), but will return a per-processor unique number that is not actually the kernel address of these structures. This serves as a mitigation against Kernel ASLR (KASLR) information leaks from local attackers.

Finally, note that there is no way to disable PatchGuard or HyperGuard once they are enabled. However, because device-driver developers might need to make changes to a running system as part of debugging, PatchGuard is not enabled when the system boots in debugging mode with an active remote kernel-debugging connection. Similarly, HyperGuard is disabled if the hypervisor boots in debugging mode with a remote debugger attached.

### **CONCLUSION**

Windows provides an extensive array of security functions that meet the key requirements of both government agencies and commercial installations. In this chapter, we’ve taken a brief tour of the internal components that are the basis of these security features. In Chapter 8 of Part 2, we’ll look at various mechanisms that are spread out throughout the Windows system.

## **Index**

### **SYMBOLS**

! (exclamation points), [40](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_40)

.NET Framework, [6](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_6)–[7](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_7)

64-bit address space layouts, [357](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_357)–[359](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_359)

64-bit extended systems, [50](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_50)

### **A**

AAM (Admin Approval Mode), [729](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_729)

access

access checks, [621](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_621)–[624](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_624)

access masks, [624](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_624)

access tokens, [20](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_20), [677](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_677)

ACEs (access control entries). *See* [ACEs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_02)

ACLs (access control lists). *See* [ACLs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_03)

object access auditing, [679](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_679)–[681](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_681)

access checks, [621](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_621)–[624](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_624)

access control entries. *See* [ACEs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_02)

access control lists. *See* [ACLs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_03)

access masks, [624](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_624)

access tokens, [20](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_20), [677](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_677)

accounting (quantums), [233](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_233)

accounts

privileges, [668](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_668)–[675](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_675)

Bypass Traverse Checking privilege, [675](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_675)

super privileges, [675](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_675)–[676](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_676)

rights, [668](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_668)–[670](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_670)

User Account Control. *See* [UAC](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_68)

ACEs (access control entries)

conditional ACEs, [667](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_667)–[668](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_668)

GUI security editors, [664](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_664)–[665](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_665)

overview, [650](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_650)–[653](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_653)

trust SIDs, [657](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_657)–[658](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_658)

ACLs (access control lists)

assigning, [656](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_656)–[657](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_657)

determining access, [659](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_659)–[665](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_665)

GUI security editors, [664](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_664)–[665](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_665)

inheritance, [656](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_656)–[657](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_657)

overview, [650](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_650)–[653](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_653)

Owner Rights SIDs, [662](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_662)

activation contexts, [163](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_163)

address spaces

64-bit address space layouts, [357](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_357)–[359](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_359)

ARM address space layouts, [356](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_356)–[357](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_357)

canonical addresses, [359](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_359)

creating processes, [140](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_140)–[142](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_142)

dynamic allocation, [359](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_359)–[365](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_365)

image bias, [368](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_368)

PTEs, [355](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_355)–[356](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_356)

quotas, [364](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_364)–[365](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_365)

sessions, [353](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_353)–[355](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_355)

setting address limits, [363](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_363)–[364](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_364)

types of data, [348](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_348)–[349](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_349)

user address spaces. *See* [user address spaces](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_70)

viewing address usage, [361](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_361)–[363](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_363)

x64 virtual address limitations, [359](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_359)

x86 address space layouts, [349](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_349)–[352](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_352)

x86 session space, [353](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_353)–[355](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_355)

x86 system address space layouts, [352](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_352)–[353](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_353)

address translation

ARM virtual address translation, [381](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_381)–[382](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_382)

overview, [371](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_371)

page tables, [375](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_375)–[376](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_376)

PTEs, [375](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_375)–[376](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_376)

TLB, [377](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_377)–[378](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_378)

viewing, [378](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_378)–[380](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_380)

write bits, [376](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_376)–[377](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_377)

x64 virtual address translation, [380](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_380)–[381](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_381)

x86 virtual address translation, [371](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_371)–[375](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_375)

Address Windowing Extensions (AWE), [22](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_22), [323](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_323)–[324](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_324)

addresses, canonical, [359](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_359)

Admin Approval Mode (AAM), [729](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_729)

administrative rights (UAC), [729](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_729)–[732](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_732)

advanced audit policy, [683](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_683)–[684](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_684)

affinity manager, [336](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_336)

affinity masks

extended affinity masks, [276](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_276)–[277](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_277)

symmetric multiprocessing, [53](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_53)

threads, [275](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_275)–[277](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_277)

allocating

address spaces

dynamic allocation, [359](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_359)–[365](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_365)

quotas, [364](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_364)–[365](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_365)

memory, [310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)–[315](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_315)

API Sets (image loader), [173](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_173)–[176](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_176)

APIs (application programming interfaces)

API Sets (image loader), [173](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_173)–[176](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_176)

AuthZ, [666](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_666)–[667](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_667)

COM (component object model), [5](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_5)

.NET Framework, [6](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_6)–[7](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_7)

overview, [4](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_4)

Windows Runtime, [5](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_5)–[6](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_6)

AppContainers

brokers, [709](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_709)

capabilities, [699](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_699)–[703](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_703)

handles, [705](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_705)–[708](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_708)

lowboxes, defined, [134](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_134)

object namespaces, [703](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_703)–[705](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_705)

overview, [684](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_684)

security environment. *See* security environment

tokens, [690](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_690)–[692](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_692)

UWP apps, [685](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_685)–[687](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_687)

UWP processes, [687](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_687)–[692](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_692)

AppIDs, [756](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_756)–[757](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_757)

application programming interfaces. *See* [APIs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_07)

applications. *See also* [processes](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_48)

APIs. *See* [APIs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_07)

AppContainers. *See* [AppContainers](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_08)

AppIDs, [756](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_756)–[757](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_757)

AppLocker, [757](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_757)–[762](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_762)

classic apps, [103](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_103)

desktop apps, [103](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_103)

immersive apps, [103](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_103)

large address spaces, [351](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_351)

modern apps, [103](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_103)

UWP apps, [685](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_685)–[687](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_687)

AppLocker, [757](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_757)–[762](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_762)

applying priority boosts, [249](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_249)

architecture

components, [61](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_61)–[62](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_62)

kernel mode, [47](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_47)–[49](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_49)

overview, [47](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_47)–[49](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_49), [61](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_61)–[62](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_62)

user mode, [47](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_47)–[49](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_49)

VBS, [59](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_59)–[61](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_61)

ARM address space layouts, [356](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_356)–[357](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_357)

ARM virtual address translation, [381](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_381)–[382](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_382)

ASLR. *See* [address spaces](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_04)

assertions

compilers, [753](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_753)

fast fail failure codes, [754](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_754)–[756](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_756)

operating system, [753](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_753)

overview, [752](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_752)–[753](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_753)

assigning

ACLs, [656](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_656)–[657](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_657)

processors (groups), [271](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_271)–[273](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_273)

assured authentication, [718](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_718)–[719](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_719)

asymmetric multiprocessing, [51](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_51)

asynchronous I/O, [511](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_511)

atom tables, [697](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_697)–[698](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_698)

attributes

AppContainer security, [695](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_695)–[697](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_697)

converting, [131](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_131)–[135](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_135)

trustlets, [125](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_125)

auditing (security)

advanced audit policy, [683](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_683)–[684](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_684)

global audit policy, [682](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_682)–[683](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_683)

object access auditing, [679](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_679)–[681](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_681)

overview, [677](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_677)–[679](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_679)

authentication

policies (Credential Guard), [616](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_616)–[617](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_617)

users, [713](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_713)–[718](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_718)

Kerberos, [714](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_714)–[715](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_715)

MSV1\_0, [713](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_713)–[714](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_714)

viewing active logon sessions, [715](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_715)–[717](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_717)

AuthZ API, [666](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_666)–[667](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_667)

Autoboost, [254](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_254)

auto-elevation (UAC), [732](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_732)–[733](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_733)

AWE (Address Windowing Extensions), [22](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_22), [323](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_323)–[324](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_324)

### **B**

balance set manager

priority boosts, [247](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_247)

working sets, [421](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_421)–[422](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_422)

bandwidth, reserving, [551](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_551)–[552](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_552)

binary planting, [160](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_160)

bitmaps (CFG), [744](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_744)–[747](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_747)

BNO isolation, [708](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_708)

brokers (AppContainers), [709](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_709)

buckets (heaps), [335](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_335)

built-in trustlets, [125](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_125)

bumps, [549](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_549), [551](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_551)

bus drivers, [493](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_493)

Bypass Traverse Checking Privilege, [675](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_675)

### **C**

caching files, [513](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_513)

calculating load address, [368](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_368)–[369](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_369)

cancelling I/O

overview, [537](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_537)

thread termination, [539](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_539)

users, [537](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_537)–[538](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_538)

canonical addresses, [359](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_359)

catalog files (Plug and Play), [574](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_574)

CBAC (Claims-Based Access Control), [667](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_667)

CC (Common Criteria), [607](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_607)

CFG (Control Flow Guard), [741](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_741)

bitmaps, [744](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_744)–[747](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_747)

image loader, [750](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_750)–[751](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_751)

kernel CFG, [751](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_751)–[752](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_752)

overview, [741](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_741)–[742](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_742)

strengthening, [747](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_747)–[749](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_749)

suppression, [741](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_741), [748](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_748)

viewing, [742](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_742)–[744](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_744)

CFI (control flow integrity), [740](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_740)

checked build (kernel debugging), [57](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_57)–[58](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_58)

checking large address space support, [351](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_351)

Claims-Based Access Control (CBAC), [667](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_667)

class drivers, [494](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_494)

classic apps, [103](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_103)

classification, memory combining, [461](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_461)

clients, memory limits, [447](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_447)–[449](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_449)

clock cycles, [233](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_233)–[234](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_234)

clock interval, [232](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_232)

clustered page faults, [387](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_387)–[388](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_388)

collided page faults, [387](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_387)

COM (component object model), [5](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_5)

combining memory

classification, [461](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_461)

combined pages release, [464](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_464)–[465](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_465)

creating shared PTEs, [462](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_462)–[464](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_464)

overview, [459](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_459)–[460](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_460)

page combining, [462](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_462)

searching, [460](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_460)

viewing, [465](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_465)–[467](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_467)

command prompt windows, opening, [13](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_13)

commands

! (exclamation points), [40](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_40)

!address, [746](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_746)

!ca, [408](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_408), [411](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_411)

!cpuinfo, [233](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_233)

!dbgprint, [58](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_58)

!dd, [380](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_380)

!devnode, [562](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_562)–[563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)

!devobj, [504](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_504)–[506](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_506), [524](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_524)

!devstack, [521](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_521)

!dq, [379](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_379)

!drvobj, [504](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_504)–[506](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_506), [512](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_512), [517](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_517)

!file, [408](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_408)

!fileobj, [509](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_509)

!handle, [109](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_109), [408](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_408), [707](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_707)

!heap, [338](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_338)

!heap -i, [341](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_341)–[343](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_343)

!heap -s, [340](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_340)

!irp, [519](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_519), [522](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_522), [541](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_541)

!irpfind, [521](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_521)

!job, [180](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_180)–[182](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_182), [265](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_265), [294](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_294)

!list, [167](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_167)

!lookaside, [331](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_331)

!memusage, [410](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_410)

!numa, [270](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_270)

!object, [502](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_502)–[504](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_504)

!partition, [448](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_448)

!pcr, [77](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_77), [260](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_260)

!peb, [109](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_109)–[110](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_110), [166](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_166)

!pfn, [443](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_443)

!poolused, [328](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_328)

!popcaps, [597](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_597)

!popolicy, [598](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_598)

!prcb, [77](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_77)s

!process, [109](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_109), [180](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_180)–[181](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_181), [197](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_197), [261](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_261), [518](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_518)

!pte, [375](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_375), [379](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_379), [466](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_466)

!ready, [230](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_230)–[231](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_231)

!runas, [180](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_180)

!sd, [655](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_655)

!session, [353](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_353)

!silo, [189](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_189)

!smt, [268](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_268)

!sysptes, [355](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_355)

!teb, [201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)–[205](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_205)

!thread, [109](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_109), [197](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_197)–[199](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_199), [201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201), [261](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_261), [518](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_518)

!token, [637](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_637)

!vad, [402](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_402)

!verifier, [557](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_557)

!vm, [307](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_307)

!vm 4, [355](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_355)

!wdfkd.wdfldr, [580](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_580)

!wmiprint, [58](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_58)

!wsle, [420](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_420)

.process, [109](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_109)–[110](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_110)

~, [202](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_202), [209](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_209)

at, [8](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_8)

AuditPol, [682](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_682)

bang commands, [40](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_40)

cron, [8](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_8)

dd, [380](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_380)

docker, [190](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_190)

dt, [40](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_40)–[42](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_42), [78](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_78), [108](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_108), [182](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_182)

dump, [40](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_40)

g, [70](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_70), [265](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_265)

k, [70](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_70), [210](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_210)

lm, [112](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_112)

poolmon, [327](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_327)

powercfg /a, [592](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_592)

powercfg /h, [591](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_591)

powercfg /list, [598](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_598)

q, [43](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_43)

runas, [101](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_101)

schtacks, [8](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_8)

start, [218](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_218)–[219](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_219)

u, [71](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_71)

ver, [3](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_3)

winver, [3](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_3)

commit charge

memory manager, [313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

page faults. *See* [page faults](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_36)

commit limit

commit charge, [394](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_394)–[396](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_396)

memory manager, [313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

committed pages (private pages), [310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)–[313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

Common Criteria (CC), [607](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_607)

communication, secure, [614](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_614)–[615](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_615)

compilers, assertions, [753](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_753)

completion ports, [541](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_541)–[546](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_546)

component object model (COM), [5](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_5)

components

architecture, [61](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_61)–[62](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_62)

I/O, [483](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_483)–[488](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_488)

memory manager, [302](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_302)–[303](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_303)

security, [608](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_608)–[611](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_611)

SuperFetch, [473](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_473)–[474](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_474)

compression (memory), [449](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_449)–[456](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_456)

concurrency (threads), [542](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_542)

conditional ACEs, [667](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_667)–[668](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_668)

configuring

DFSS, [290](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_290)–[292](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_292)

quantum, [237](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_237)–[238](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_238)

Connected Standby, [594](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_594)

consent elevation (UAC), [729](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_729)

consoles, [67](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_67)

container notification (I/O), [552](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_552)

containers (silos)

ancillary functionality, [189](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_189)–[190](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_190)

containers, [190](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_190)–[191](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_191)

contexts, [186](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_186)–[188](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_188)

creating, [188](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_188)–[189](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_189)

isolation, [184](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_184)–[186](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_186)

monitors, [187](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_187)–[188](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_188)

objects, [183](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_183)–[184](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_184)

overview, [183](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_183)

context switches, [215](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_215)

context switching, [255](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_255)–[256](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_256)

contexts

activation contexts, [163](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_163)

context switches, [215](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_215)

context switching, [255](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_255)–[256](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_256)

Direct Switch, [255](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_255)–[256](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_256)

directed context switch, [19](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_19)

image loader, [163](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_163)

jobs (silos), [186](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_186)–[188](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_188)

threads, [18](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_18)

Control Flow Guard. *See* [CFG](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_12)

control flow integrity (CFI), [740](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_740)

controlling

power, [599](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_599)–[600](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_600)

quantums, [234](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_234)–[235](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_235)

converting attributes, [131](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_131)–[135](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_135)

copy-on-write, [321](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_321)–[323](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_323)

cores (threads), [52](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_52)

CPU

rate limits, [292](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_292)–[295](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_295)

sets, [278](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_278)–[283](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_283)

starvation, [246](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_246)–[248](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_248)

Credential Guard

authentication policies, [616](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_616)–[617](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_617)

Kerberos armoring, [616](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_616)–[617](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_617)

NTOWF/TGT key, [613](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_613)–[614](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_614)

overview, [612](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_612)–[613](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_613)

passwords, [613](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_613)

secure communication, [614](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_614)–[615](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_615)

UEFI, [616](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_616)

credential providers (DLLs), [98](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_98)

CSR\_PROCESS data structure, [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105), [111](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_111)–[112](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_112)

CSR\_THREAD data structure, [195](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_195), [205](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_205)–[206](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_206)

### **D**

DAC (Dynamic Access Control), [666](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_666)

DACLs. *See* [ACLs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_03)

data

address spaces, [348](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_348)–[349](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_349)

loading, [471](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_471)–[472](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_472)

Data Execution Protection (DEP), [319](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_319)–[321](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_321)

data structures

PFN, [440](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_440)–[443](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_443)

processes

!process command, [109](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_109)

CSR\_PROCESS, [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105), [111](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_111)–[112](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_112)

DXGPROCESS, [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105)

EPROCESS, [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105)–[108](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_108)

ETHREAD, [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105)

KPROCESS, [106](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_106)–[107](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_107)

W32PROCESS, [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105), [113](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_113)

threads

CSR\_THREAD, [195](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_195), [205](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_205), [206](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_206)

ETHREAD, [194](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_194)–[201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)

KTHREAD, [194](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_194)–[201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)

databases

dispatcher database, [228](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_228)–[230](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_230)

image loader, [164](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_164)–[168](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_168)

loaded modules database, [164](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_164)–[168](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_168)

deadline scheduling, [254](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_254)

debuggers

kernel-mode debugger, [210](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_210)–[212](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_212)

user-mode debugger, [209](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_209)–[210](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_210)

debugging

DebugActiveProcess function, [39](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_39)

DebugBreak function, [194](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_194)

heaps, [342](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_342)–[346](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_346)

kernel-mode debugger, [210](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_210)–[212](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_212)

kernel debugging. *See* kernel debugging

unkillable processes, [539](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_539)–[541](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_541)

user-mode debugger, [209](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_209)–[210](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_210)

Debugging Tools for Windows (kernel debugging), [38](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_38)–[42](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_42)

kernel-mode debugging, [39](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_39)–[40](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_40)

user-mode debugging, [39](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_39)

viewing type information, [41](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_41)–[42](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_42)

Deferred Procedure Calls (DPCs)

I/O, [490](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_490)–[492](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_492)

stacks, [401](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_401)

demand paging, [413](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_413)

DEP (Data Execution Protection), [319](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_319)–[321](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_321)

Dependency Walker

exported functions, [34](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_34)–[35](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_35)

HAL image dependencies, [80](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_80)–[82](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_82)

subsystems, [62](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_62)–[63](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_63)

design goals (Windows), [45](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_45)–[46](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_46)

desktop apps, [103](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_103)

determining access (ACLs), [659](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_659)–[665](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_665)

device drivers

bus drivers, [493](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_493)

class drivers, [494](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_494)

device objects, [500](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_500)–[507](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_507)

dispatch routines, [504](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_504), [517](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_517)–[518](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_518)

driver objects, [500](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_500)–[507](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_507)

Driver Verifier. *See* [Driver Verifier](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_18)

file objects, [507](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_507)–[510](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_510)

filter drivers, [493](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_493)

function drivers, [493](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_493)

installing, [571](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_571)–[575](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_575), [577](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_577)

IRPs

dispatch routines, [517](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_517)–[518](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_518)

layered drivers, [533](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_533)–[536](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_536)

overview, [525](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_525)–[528](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_528)

user address spaces, [528](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_528)–[531](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_531)

KMDF, [578](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_578)–[587](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_587)

layered drivers

IRPs, [533](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_533)–[536](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_536)

overview, [494](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_494)–[496](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_496)

loading, [575](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_575)–[577](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_577)

opening devices, [507](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_507)–[510](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_510)

overview, [82](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_82)–[83](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_83), [492](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_492)–[496](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_496)

port drivers, [494](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_494)–[495](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_495)

power manager, [596](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_596)

routines, [498](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_498)–[500](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_500)

support, [560](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_560)–[561](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_561), [569](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_569)–[571](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_571)

types, [492](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_492)–[496](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_496)

UMDF, [578](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_578)–[581](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_581), [587](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_587)–[590](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_590)

Universal Windows Drivers, [85](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_85)

viewing, [85](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_85)–[88](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_88), [496](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_496)–[498](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_498)

WDF. *See* [WDF](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_76)

WDK (Windows Driver Kit), [43](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_43)–[44](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_44)

WDM, [83](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_83)–[84](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_84), [493](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_493)–[494](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_494)

Device Guard, [617](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_617)–[619](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_619)

device objects (device drivers), [500](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_500)–[507](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_507)

devices

drivers. *See* [device drivers](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_16)

device stacks, [563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)–[569](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_569)

devnodes, [563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)–[569](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_569)

enumeration, [561](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_561)–[563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)

opening, [507](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_507)–[510](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_510)

support, [560](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_560)–[561](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_561)

trees, [561](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_561)–[563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)

DFSS (dynamic fair share scheduling), [289](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_289)–[292](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_292)

Direct Swit0ch, [255](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_255)–[256](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_256)

directed context switch, [19](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_19)

dispatch events, [239](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_239)–[240](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_240)

dispatch routines

device drivers, [504](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_504)

IRPs, [517](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_517)–[518](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_518)

dispatcher database, [228](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_228)–[230](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_230)

dispatchers, [215](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_215)

displaying. *See* [viewing](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_74)

DLLs

credential providers, [98](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_98)

DllMain function, [154](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_154)

image loader

import parsing, [168](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_168)–[170](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_170)

name redirection, [162](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_162)–[163](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_163)

name resolution, [160](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_160)–[162](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_162)

safe DLL search mode, [160](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_160)

viewing DLL load search order, [163](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_163)–[164](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_164)

Ntdll.dll, [70](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_70)–[72](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_72)

overview, [8](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_8)

subsystem DLLs, [48](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_48)

subsystems, [62](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_62)–[63](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_63)

DPCs (Deferred Procedure Calls)

I/O, [490](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_490)–[492](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_492)

stacks, [401](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_401)

driver objects (device drivers), [500](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_500)–[507](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_507)

Driver Verifier

I/O, [554](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_554)–[555](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_555)

memory

IRQL checking, [557](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_557)

low resources simulation, [557](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_557)–[558](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_558)

miscellaneous checks, [558](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_558)–[559](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_559)

pool tracking, [556](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_556)–[557](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_557)

special pool, [555](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_555)–[556](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_556)

overview, [552](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_552)–[554](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_554)

drivers. *See* [device drivers](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_16)

dumping

device trees, [562](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_562)–[563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)

dump command, [40](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_40)

ETHREAD structure, [197](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_197)–[198](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_198)

KTHREAD structure, [197](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_197)–[198](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_198)

silo contexts, [187](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_187)–[188](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_188)

DXGPROCESS data structure, [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105)

Dynamic Access Control (DAC), [666](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_666)

dynamic allocation (address spaces), [359](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_359)–[365](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_365)

dynamic fair share scheduling (DFSS), [289](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_289)–[292](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_292)

dynamic processors, [295](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_295)–[296](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_296)

### **E**

editions (Windows), [54](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_54)–[57](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_57)

EMET (Enhanced Mitigation Experience Toolkit), [370](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_370)

enclaves (memory), [467](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_467)–[472](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_472)

Enhanced Mitigation Experience Toolkit (EMET), [370](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_370)

entries (PFN), [443](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_443)

enumeration (devices), [561](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_561)–[563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)

environment (AppContainers security)

overview, [693](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_693)–[695](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_695)

viewing atom table, [697](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_697)–[698](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_698)

viewing security attributes, [695](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_695)–[697](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_697)

EPROCESS data structure, [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105)–[108](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_108)

EPROCESS object, [138](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_138)–[140](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_140)

ETHREAD data structure, [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105), [194](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_194)–[201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)

events

dispatch events, [239](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_239)–[240](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_240)

notification events, [423](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_423)–[425](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_425)

examining. *See* [viewing](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_74)

exclamation points (!), [40](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_40)

executing initial thread, [148](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_148)

executive, [72](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_72)–[75](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_75)

executive process object, [138](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_138)–[143](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_143)

executive resources (priority boosts), [242](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_242)–[243](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_243)

exiting threads, [260](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_260)

experiments

allocating memory, [311](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_311)–[313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

Bypass Traverse Checking Privilege, [675](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_675)

calculating load address, [368](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_368)–[369](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_369)

checked build, [57](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_57)–[58](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_58)

checking large address space support, [351](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_351)

configuring quantums, [237](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_237)–[238](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_238)

creating maximum number of threads, [399](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_399)

debugging unkillable processes, [539](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_539)–[541](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_541)

dumping

device trees, [562](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_562)–[563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)

ETHREAD structure, [197](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_197)–[198](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_198)

KTHREAD structure, [197](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_197)–[198](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_198)

silo contexts, [187](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_187)–[188](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_188)

identifying trustlets, [129](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_129)

launching programs at low integrity levels, [641](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_641)–[642](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_642)

Performance Monitor kernel mode/user mode comparison, [26](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_26)–[27](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_27)

setting address limits, [363](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_363)–[364](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_364)

tracing process startup, [149](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_149)–[154](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_154)

troubleshooting pool leaks, [329](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_329)–[330](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_330)

using virtual service accounts, [647](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_647)–[650](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_650)

viewing

access masks, [624](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_624)

active logon sessions, [715](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_715)–[717](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_717)

address translation, [378](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_378)–[380](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_380)

address usage, [361](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_361)–[363](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_363)

AppContainer atom table, [697](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_697)–[698](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_698)

AppContainer capabilities, [701](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_701)–[703](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_703)

AppContainer security attributes, [695](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_695)–[697](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_697)

AppContainer tokens, [690](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_690)–[692](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_692)

brokers, [709](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_709)

CFG, [742](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_742)–[744](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_744)

CFG bitmaps, [746](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_746)–[747](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_747)

clock cycles per quantum, [233](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_233)–[234](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_234)

control areas, [408](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_408)–[412](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_412)

CPU rate limits, [293](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_293)–[295](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_295)

CPU sets, [279](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_279)–[283](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_283)

CPU starvation, [247](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_247)–[248](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_248)

CSR\_PROCESS structure, [112](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_112)

CSR\_THREAD structure, [206](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_206)

DEP, [321](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_321)

device drivers, [85](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_85)–[88](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_88)

device objects, [502](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_502)–[506](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_506)

devnodes, [568](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_568)–[569](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_569)

DLL load search order, [163](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_163)–[164](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_164)

driver catalog files, [574](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_574)

driver dispatch routines, [517](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_517)–[518](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_518)

driver INF files, [573](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_573)

driver objects, [504](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_504)–[506](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_506)

driver power mappings, [596](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_596)

drivers, [496](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_496)–[498](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_498)

DSS, [290](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_290)–[292](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_292)

enabling privileges, [673](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_673)–[675](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_675)

EPROCESS data structure, [107](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_107)–[108](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_108)

ETHREAD structure, [196](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_196)–[201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)

exported functions, [34](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_34)–[35](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_35)

fast I/O, [512](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_512)–[513](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_513)

file objects, [508](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_508)–[509](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_509)

file virtualization, [726](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_726)–[727](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_727)

filtered admin tokens, [645](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_645)–[646](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_646)

foreground boosts, [243](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_243)–[245](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_245)

free page lists, [429](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_429)–[430](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_430)

global audit policy, [682](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_682)

GUI boosts, [245](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_245)–[246](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_246)

HAL image dependencies, [80](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_80)–[82](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_82)

heaps, [338](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_338)–[341](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_341)

idle threads, [260](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_260)–[262](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_262)

image loader, [156](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_156)–[157](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_157)

integrity levels, [628](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_628)–[631](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_631)

I/O priority boosting/bumping, [551](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_551)

I/O priority throughput, [549](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_549)–[551](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_551)

IRPs, [518](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_518)–[519](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_519), [521](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_521)–[524](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_524)

jobs, [180](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_180)–[183](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_183)

kernel stacks, [400](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_400)–[401](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_401)

kernel type information, [41](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_41)–[42](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_42)

KMDF drivers, [580](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_580)–[581](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_581)

KPCR, [77](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_77)–[78](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_78)

KPRCB, [77](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_77)–[78](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_78)

KTHREAD structure, [196](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_196)–[201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)

loaded modules database, [166](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_166)–[167](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_167)

look-aside lists, [331](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_331)–[332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

memory, [305](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_305)–[308](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_308)

memory compression, [455](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_455)–[456](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_456)

memory partitions, [458](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_458)

memory-mapped files, [316](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_316)–[317](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_317)

MMCSS priority boosting, [252](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_252)–[253](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_253)

modified page lists, [430](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_430)–[435](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_435)

notification events, [424](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_424)–[425](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_425)

NUMA processors, [270](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_270)–[271](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_271)

object access auditing, [679](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_679)–[681](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_681)

page files, [390](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_390)–[392](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_392), [397](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_397)–[398](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_398)

page priorities, [437](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_437)

pageheap, [344](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_344)–[346](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_346)

PEB, [110](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_110)–[111](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_111)

PFN, [427](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_427)–[428](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_428)

PFN entries, [443](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_443)

pool size, [326](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_326)–[327](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_327)

power availability requests, [603](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_603)

power states, [592](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_592)–[593](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_593)

prefetch file reads and writes, [415](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_415)–[416](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_416)

processes, affinity, [275](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_275)–[276](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_276)

processes, data structures, [109](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_109)

processes, process tree, [12](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_12)–[13](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_13)

processes, Process Explorer, [16](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_16)–[18](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_18)

processes, Task Manager, [9](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_9)–[11](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_11)

protected processes, [118](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_118)–[119](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_119)

PTEs, [355](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_355)–[356](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_356)

ready threads, [230](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_230)–[231](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_231)

section objects, [406](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_406)–[407](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_407)

security descriptors, [654](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_654)–[656](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_656)

service processes, [97](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_97)–[98](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_98)

services, [97](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_97)

sessions, [353](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_353)–[355](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_355)

SIDs, [626](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_626)–[627](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_627)

SMT processors, [268](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_268)–[269](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_269)

SRPs, [764](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_764)

stacks, [521](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_521)

standby page lists, [430](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_430)–[435](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_435)

subsystems, [62](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_62)–[63](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_63)

swap files, [393](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_393)

system power capabilities, [597](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_597)–[599](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_599)

system service dispatcher, [70](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_70)–[71](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_71)

TEB, [201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)–[205](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_205)

thread freezing, [265](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_265)–[266](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_266)

thread pools, [299](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_299)–[300](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_300)

thread priorities, [219](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_219)–[222](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_222)

thread states, [224](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_224)–[228](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_228)

threads, clock interval, [232](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_232)

threads, kernel-mode debugger, [210](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_210)–[212](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_212)

threads, protected processes, [213](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_213)

threads, user-mode debugger, [209](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_209)–[210](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_210)

token stored handles, [706](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_706)–[708](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_708)

tokens, [635](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_635)–[640](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_640)

trust SIDs, [658](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_658)

UMDF drivers, [580](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_580)–[581](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_581)

user address spaces, [366](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_366)–[367](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_367)

UWP processes, [689](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_689)–[690](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_690)

VADs, [402](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_402)–[403](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_403)

virtual page files, [393](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_393)

Windows edition enabled features, [56](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_56)–[57](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_57)

working sets, [418](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_418)–[421](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_421)

zero page lists, [429](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_429)–[430](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_430)

exploit mitigation

assertions. *See* [assertions](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_10)

CFG. *See* [CFG](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_12)

control flow integrity, [740](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_740)

overview, [735](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_735)

process mitigation policies, [735](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_735)–[740](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_740)

extended affinity masks, [276](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_276)–[277](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_277)

### **F**

facilities (Windows edition enabled features), [56](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_56)–[57](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_57)

fast fail failure codes, [754](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_754)–[756](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_756)

fast I/O, [511](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_511)–[513](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_513)

fast user switching, [30](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_30), [475](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_475)

fault-tolerant heaps (FTH), [347](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_347)–[348](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_348)

fibers, [19](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_19)

file mapping objects, [20](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_20)

file objects (device driv0ers), [507](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_507)–[510](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_510)

files

caching, [513](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_513)

catalog files, [574](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_574)

file mapping objects, [20](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_20)

INF files, [573](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_573)

mapped-file I/O, [513](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_513)

memory-mapped files, [315](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_315)–[317](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_317)

page files. *See* [page files](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_37)

virtualization (UAC), [722](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_722)–[727](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_727)

filter drivers, [493](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_493)

filtered admin tokens (SIDs), [645](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_645)–[646](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_646)

filters (function drivers), [493](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_493)

firmware, [29](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_29)

flags, converting, [131](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_131)–[135](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_135)

flow (IRPs), [519](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_519)–[520](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_520)

foreground threads, [243](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_243)–[245](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_245)

frameworks

.NET Framework, [6](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_6)–[7](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_7)

power management framework, [600](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_600)–[601](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_601)

WBF (Windows Biometric Framework), [719](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_719)–[721](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_721)

free pages

lists, [429](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_429)–[430](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_430)

memory manager, [310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)–[313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

freezing threads, [264](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_264)–[266](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_266)

FTH (fault-tolerant heaps), [347](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_347)–[348](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_348)

functions

AllocConsole, [63](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_63)

AvTaskIndexYield, [254](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_254)

BaseThreadInit, [160](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_160), [170](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_170)

ConvertThreadToFiber, [19](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_19)

CreateFiber, [19](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_19)

CreateFile, [34](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_34)

CreateProcess, [101](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_101)–[104](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_104), [129](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_129)–[131](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_131), [134](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_134), [157](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_157)

CreateProcessAsUser, [101](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_101)–[103](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_103), [139](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_139)

CreateProcessInternal, [101](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_101)–[103](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_103)

CreateProcessInternalW, [129](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_129), [131](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_131), [134](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_134)–[138](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_138), [146](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_146)–[147](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_147), [150](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_150)

CreateProcessWithLogonW, [101](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_101)–[103](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_103)

CreateProcessWithTokenW, [101](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_101)–[103](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_103)

CreateRemoteThread, [193](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_193)–[194](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_194)

CreateRemoteThreadEx, [194](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_194), [206](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_206)–[207](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_207)

CreateThread, [193](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_193)–[194](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_194), [199](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_199), [208](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_208)

Csr, [71](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_71)–[72](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_72)

DbgUi, [72](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_72)

DebugActiveProcess, [39](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_39)

DebugBreak, [194](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_194)

DeviceIoControl, [73](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_73)

DgbPrintEx, [58](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_58)

DllMain, [154](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_154)

drivers, [493](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_493)

Etw, [72](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_72)

Ex, [73](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_73)

ExitProcess, [154](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_154)

exported, [34](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_34)–[35](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_35)

GetQueueCompletionStatus, [176](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_176)

GetSystemTimeAdjustment, [232](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_232)

GetThreadContext, [18](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_18)

GetVersionEx, [2](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_2)

HeapAlloc, [332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

HeapCreate, [332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

HeapDestroy, [332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

HeapFree, [332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

HeapLock, [332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

HeapReAlloc, [332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

HeapUnlock, [332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

HeapWalk, [332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

Inbv, [73](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_73)

Io, [73](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_73)

IoCompleteRequest, [241](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_241)

Iop, [73](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_73)

Ke, [75](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_75)

KeStartDynamicProcessor, [295](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_295)

KiConvertDynamicHeteroPolicy, [287](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_287)

KiDeferredReadyThread, [274](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_274), [284](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_284)

KiDirectSwitchThread, [256](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_256)

KiExitDispatcher, [241](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_241), [256](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_256)

KiProcessDeferredReadyList, [274](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_274)

KiRemoveBoostThread, [241](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_241), [250](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_250)

KiSearchForNewThreadOnProcessor, [283](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_283)

KiSelecthreadyThreadEx, [267](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_267)

KiSelectNextThread, [266](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_266)–[267](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_267)

Ldr, [70](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_70)–[71](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_71)

LdrApplyFileNameRedirection, [175](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_175)

Mi, [73](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_73)

MiZeroInParallel, [303](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_303)

NtCreateProcessEx, [104](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_104), [120](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_120)

NtCreateThreadEx, [207](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_207)

NtCreateUserProcess, [104](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_104)

NtCreateWorkerFactory, [298](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_298)

OpenProcess, [39](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_39)

overview, [7](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_7)–[8](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_8)

PopInitializeHeteroProcessors, [286](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_286)

prefixes, list of, [87](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_87)–[88](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_88)

PsCreateSystemThread, [194](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_194)

PspAllocateProcess, [104](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_104), [138](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_138)

PspComputeQuantum, [235](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_235)

PspCreatePicoProcess, [104](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_104)

PspInitializeApiSetMap, [175](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_175)

PspInsertProcess, [104](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_104), [143](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_143)

PsTerminateSystemThread, [194](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_194)

QueryInformationJobObject, [179](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_179)

ReadFile, [25](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_25)

ReadProcessMemory, [20](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_20)

ResumeThread, [264](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_264)

Rtl, [72](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_72)

RtlAssert, [58](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_58)

RtlCreateUserProcess, [104](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_104)

RtlGetVersion, [55](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_55)

RtlUserThreadStart, [208](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_208)

RtlVerifyVersionInfo, [55](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_55)

secure system calls, [71](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_71)

SetInformationJobObject, [275](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_275)

SetPriorityClass, [218](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_218)

SetProcessAffinityMask, [275](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_275)

SetProcessWorkingSetSize, [222](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_222)

SetThreadAffinityMask, [275](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_275)

SetThreadIdealProcessor, [278](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_278)

SetThreadSelectedCpuSets, [279](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_279)

subsystem DLLs, [63](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_63)

SuspendThread, [264](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_264)

SwitchToFiber, [19](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_19)

system services, [72](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_72)

SystemParametersInfo, [178](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_178)

TerminateJobObject, [179](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_179)

TerminateProcess, [154](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_154)

TermsrvGetWindowsDirectoryW, [170](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_170)

TimeBeginPeriod, [232](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_232)

TimeSetEvent, [232](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_232)

TpAllocJobNotification, [176](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_176)

UserHandleGrantAccess, [176](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_176)

VerifyVersionInfo, [2](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_2), [55](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_55)

VirtualLock, [314](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_314)

WaitForMultipleObjects, [256](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_256)

WaitForSingleObject, [256](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_256)

Wow64GetThreadContext, [19](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_19)

WriteProcessMemory, [20](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_20)

ZwUserGetMessage, [210](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_210)

### **G**

games (priority boosts), [251](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_251)–[254](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_254)

global audit policy, [682](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_682)–[683](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_683)

granularity (memory), [314](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_314)–[315](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_315)

groups

claims, [718](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_718)

processors, [271](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_271)–[273](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_273)

scheduling

CPU rate limits, [292](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_292)–[295](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_295)

DFSS, [289](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_289)–[292](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_292)

dynamic processors, [295](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_295)–[296](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_296)

overview, [287](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_287)–[289](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_289)

GUI

security editors, [664](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_664)–[665](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_665)

threads (priority boosts), [245](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_245)–[246](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_246)

GUIDs (Switchback), [171](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_171)–[173](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_173)

### **H**

HAL (hardware abstraction layer)

overview, [79](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_79)–[82](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_82)

viewing image dependencies, [80](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_80)–[82](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_82)

handles

AppContainers, [705](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_705)–[708](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_708)

token stored handles, [706](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_706)–[708](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_708)

hardware

firmware, [29](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_29)

HAL. *See* [HAL](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_21)

kernel support, [78](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_78)–[79](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_79)

hashes (AppContainer atom tables), [697](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_697)–[698](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_698)

heaps/pools

affinity manager, [336](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_336)

buckets, [335](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_335)

debugging, [342](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_342)–[346](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_346)

FTH (fault tolerant heaps), [347](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_347)–[348](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_348)

HeapAlloc function, [332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

HeapCreate function, [332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

HeapDestroy function, [332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

HeapFree function, [332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

HeapLock function, [332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

HeapReAlloc function, [332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

HeapUnlock function, [332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

HeapWalk function, [332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

leaks, [329](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_329)–[330](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_330)

LFH (low-fragmentation heaps), [335](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_335)–[336](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_336)

look-aside lists, [331](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_331)–[332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

NT heaps, [334](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_334)

non-paged pools, [325](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_325)

overview, [324](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_324)–[325](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_325), [332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

paged pools, [325](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_325)

pageheap, [343](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_343)–[346](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_346)

poolmon command, [327](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_327)

processes, [333](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_333)

randomizing (user address spaces), [369](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_369)

security, [341](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_341)–[342](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_342)

segment heaps, [336](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_336)–[337](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_337)

size, [325](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_325)–[327](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_327)

special pools, [325](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_325), [555](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_555)–[556](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_556)

synchronization, [334](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_334)–[335](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_335)

thread pools, [297](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_297)–[300](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_300)

tracking, [556](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_556)–[557](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_557)

types, [334](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_334)

usage, [327](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_327)–[329](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_329)

viewing, [338](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_338)–[341](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_341)

heterogenous multiprocessing, [52](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_52)

heterogenous scheduling, [286](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_286)–[287](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_287)

hibernation, [475](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_475)

hiding. *See* [viewing](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_74)

hierarchies (jobs), [179](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_179)–[180](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_180)

host (consoles), [67](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_67)

hybrid jobs, [183](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_183)

HyperGuard, [768](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_768)–[769](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_769)

hypervisor, [27](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_27)–[28](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_28)

### **I**

IBAC (Identity-Based Access Control), [667](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_667)

ideal node, [278](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_278)

ideal processor, [277](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_277)–[278](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_278)

identification

AppIDs, [756](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_756)–[757](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_757)

trustlets, [129](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_129)

identities (trustlets), [125](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_125)–[126](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_126)

Identity-Based Access Control (IBAC), [667](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_667)

Idle process, [89](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_89)–[90](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_90)

idle threads, [260](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_260)–[263](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_263), [267](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_267)

image bias, [368](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_368)

image loader

activation contexts, [163](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_163)

API Sets, [173](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_173)–[176](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_176)

binary planting, [160](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_160)

CFG, [750](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_750)–[751](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_751)

DLL import parsing, [168](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_168)–[170](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_170)

DLL load search order, [163](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_163)–[164](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_164)

DLL name redirection, [162](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_162)–[163](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_163)

DLL name resolution, [160](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_160)–[162](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_162)

early initialization, [157](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_157)–[160](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_160)

loaded modules database, [164](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_164)–[168](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_168)

overview, [155](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_155)–[156](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_156)

post-import process initialization, [170](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_170)

safe DLL search mode, [160](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_160)

Switchback, [171](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_171)–[173](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_173)

viewing, [156](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_156)–[157](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_157)

images

HAL, [80](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_80)–[82](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_82)

image bias, [368](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_368)

loading. *See* [image loader](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_24)

native images (subsystems), [72](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_72)

opening, [135](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_135)–[138](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_138)

randomizing (user address spaces), [367](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_367)–[369](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_369)

immersive applications, [103](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_103)

immersive processes, [103](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_103)–[104](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_104)

impersonation (SIDs), [642](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_642)–[644](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_644)

implementation (SAS), [712](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_712)

importing DLLs, [168](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_168)–[170](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_170)

INF files, [573](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_573)

inheritance (ACLs), [656](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_656)–[657](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_657)

initial thread

creating, [144](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_144)–[146](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_146)

executing, [148](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_148)

initializing

memory enclaves, [469](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_469), [472](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_472)

processes

creating processes, [148](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_148)–[149](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_149)

image loader, early, [157](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_157)–[160](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_160)

image loader, post-import process, [170](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_170)

subsystem, [146](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_146)–[147](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_147)

Winlogon, [711](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_711)–[712](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_712)

in-paging I/O, [386](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_386)–[387](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_387)

input. *See* [I/O](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_23)

installing drivers, [571](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_571)–[575](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_575), [577](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_577)

integrity levels (SIDs), [628](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_628)–[631](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_631), [641](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_641)–[642](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_642)

interfaces (APIs)

API Sets (image loader), [173](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_173)–[176](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_176)

AuthZ, [666](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_666)–[667](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_667)

COM (component object model), [5](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_5)

.NET Framework, [6](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_6)–[7](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_7)

overview, [4](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_4)

Windows Runtime, [5](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_5)–[6](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_6)

internal synchronization, [308](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_308)

Interrupt Request Levels (IRQLs), [488](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_488)–[490](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_490), [557](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_557)

invalid PTEs, [384](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_384)–[385](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_385)

inversion

I/O priorities, [549](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_549)

priority boosts, [246](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_246)

I/O

asynchronous, [511](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_511)

cancelling

overview, [537](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_537)

thread termination, [539](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_539)

users, [537](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_537)–[538](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_538)

completion ports, [541](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_541)–[546](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_546)

components, [483](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_483)–[488](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_488)

concurrency, [542](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_542)

container notifications, [552](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_552)

device drivers. *See* [device drivers](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_16)

DPCs, [490](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_490)–[492](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_492)

Driver Verifier, [554](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_554)–[555](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_555), [557](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_557)

fast I/O, [511](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_511)–[513](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_513)

file caching, [513](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_513)

in-paging I/O, [386](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_386)–[387](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_387)

I/O manager, [485](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_485)–[486](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_486)

IRPs (I/O request packets). See [IRPs (I/O request packets)](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_26)

IRQLs, [488](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_488)–[490](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_490), [557](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_557)

mapped-file I/O, [513](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_513)

overview, [483](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_483)–[488](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_488)

Plug and Play

catalog files, [574](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_574)

device enumeration, [561](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_561)–[563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)

device stacks, [563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)–[569](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_569)

device support, [560](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_560)–[561](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_561)

device trees, [561](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_561)–[563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)

devnodes, [563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)–[569](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_569)

driver installation, [571](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_571)–[575](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_575)

driver support, [560](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_560)–[561](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_561), [569](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_569)–[571](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_571)

INF files, [573](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_573)

overview, [559](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_559)–[560](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_560)

power manager

controlling power, [599](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_599)–[600](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_600)

drivers, [596](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_596)

overview, [590](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_590)–[594](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_594)

performance states, [601](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_601)

power availability requests, [602](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_602)–[603](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_603)

power management framework, [600](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_600)–[601](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_601)

power mappings, [595](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_595)

power states, [590](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_590)–[594](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_594)

system capabilities, [597](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_597)–[599](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_599)

priorities. *See* [priorities](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_46)

priority boosts, [241](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_241)–[242](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_242)

processing, [486](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_486)–[488](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_488)

scatter/gather I/O, [513](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_513)

synchronous, [511](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_511)

thread-agnostic I/O, [536](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_536)–[537](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_537)

WDF. *See* [WDF](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_76)

I/O manager, [485](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_485)–[486](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_486)

I/O request packets. *See* [IRPs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_26)

IoCompletion object, [542](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_542)

IRPs (I/O request packets), [513](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_513)

cancelling 537–[539](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_539)

dispatch routines, [517](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_517)–[518](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_518)

flow, [519](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_519)–[520](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_520)

layered drivers, [533](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_533)–[536](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_536)

overview, [513](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_513)–[515](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_515), [525](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_525)–[528](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_528)

stacks, [515](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_515)–[519](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_519), [521](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_521)

synchronization, [531](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_531)–[533](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_533)

user address spaces, [528](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_528)–[531](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_531)

viewing, [518](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_518)–[519](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_519), [521](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_521)–[524](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_524)

IRQLs (Interrupt Request Levels), [488](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_488)–[490](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_490), [557](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_557)

isolation (jobs), [184](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_184)–[186](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_186)

### **J**

jobs

creating, [178](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_178)–[179](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_179)

hierarchies, [179](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_179)–[180](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_180)

hybrid jobs, [183](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_183)

limits, [177](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_177)–[178](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_178)

overview, [20](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_20)–[21](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_21), [176](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_176)–[177](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_177)

silos. *See* [silos (jobs)](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_60)

viewing, [180](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_180)–[183](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_183)

### **K**

Kerberos

armoring, [616](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_616)–[617](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_617)

authentication, [714](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_714)–[715](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_715)

kernel

debugging

checked build, [57](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_57)–[58](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_58)

Debugging Tools for Windows, [38](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_38)–[42](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_42)

kernel-mode debugging, [39](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_39)–[40](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_40)

LiveKd, [43](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_43)

overview, [38](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_38)

symbols, [38](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_38)

threads, [210](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_210)–[212](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_212)

type information, [41](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_41)–[42](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_42)

user-mode debugging, [39](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_39)

hardware support, [78](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_78)–[79](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_79)

jobs. *See* [jobs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_27)

kernel CFG, [751](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_751)–[752](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_752)

kernel mode. *See* [kernel mode](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_29)

KPCR, [76](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_76)–[78](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_78)

KPRCB, [76](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_76)–[78](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_78)

objects, [75](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_75)

overview, [75](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_75)

patches

processes

defined, [106](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_106)

structure, [141](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_141)

secure kernel, [59](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_59)–[61](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_61)

stacks, [400](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_400)–[401](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_401)

user address spaces, [369](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_369)

kernel mode

architecture, [47](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_47)–[49](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_49)

user mode comparison, [23](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_23)–[27](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_27), [46](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_46)

kernel patches

HyperGuard, [768](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_768)–[769](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_769)

overview, [764](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_764)–[765](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_765)

PatchGuard, [765](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_765)–[768](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_768)

kernel processor control block (KPRCB), [76](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_76)–[78](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_78)

kernel processor control region (KPCR), [76](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_76)–[78](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_78)

KMDF (WDF), [578](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_578)–[587](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_587)

KPCR (kernel processor control region), [76](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_76)–[78](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_78)

KPRCB (kernel processor control block), [76](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_76)–[78](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_78)

KPROCESS data structure, [106](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_106)–[107](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_107)

KTHREAD data structure, [194](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_194)–[201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)

### **L**

large address spaces, checking support, [351](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_351)

large pages (memory manager), [303](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_303)–[304](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_304)

last processor, [277](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_277)–[278](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_278)

launching programs at low integrity levels, [641](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_641)–[642](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_642)

layered drivers

IRPs, [533](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_533)–[536](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_536)

overview, [494](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_494)–[496](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_496)

layouts

64-bit address space layouts, [357](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_357)–[359](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_359)

ARM address space layouts, [356](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_356)–[357](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_357)

user address spaces, [365](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_365)–[367](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_367)

x86 address space layouts, [349](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_349)–[352](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_352)

x86 system address space layouts, [352](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_352)–[353](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_353)

lazy evaluation, [323](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_323)

leaks (pools), [329](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_329)–[330](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_330)

Legacy Standby, [594](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_594)

levels

SID integrity levels, [628](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_628)–[631](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_631), [641](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_641)–[642](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_642)

thread priorities, [215](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_215)–[219](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_219)

LFH, [335](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_335)–[336](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_336) (low-fragmentation heaps), [335](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_335)–[336](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_336)

lightweight threads, [19](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_19)

Linux subsystems, [68](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_68)–[70](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_70)

lists

look-aside lists (pools), [331](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_331)–[332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

minimum TCB list, [117](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_117)

page lists, [429](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_429)–[435](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_435)

LiveKd, [43](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_43)

load address (user address spaces), [368](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_368)–[369](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_369)

loading

data (memory enclaves), [471](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_471)–[472](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_472)

drivers, [575](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_575)–[577](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_577)

images. *See* [image loader](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_24)

load address (user address spaces), [368](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_368)–[369](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_369)

locking/locks

memory, [314](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_314)

priority boosts, [241](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_241)

logging (SuperFetch), [474](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_474)–[475](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_475)

logical prefetcher (working sets), [413](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_413)–[416](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_416)

logon

assured authentication, [718](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_718)–[719](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_719)

group claims, [718](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_718)

overview, [710](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_710)–[711](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_711)

SAS implementation, [712](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_712)

user authentication, [713](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_713)–[718](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_718)

Kerberos, [714](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_714)–[715](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_715)

MSV1\_0, [713](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_713)–[714](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_714)

viewing active logon sessions, [715](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_715)–[717](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_717)

WBF, [719](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_719)–[721](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_721)

Windows Hello, [721](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_721)

Windows logon process, [98](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_98)–[99](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_99)

Winlogon initialization, [711](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_711)–[712](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_712)

look-aside lists, [331](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_331)–[332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

low resources simulation, [557](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_557)–[558](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_558)

LowBox. *See* [AppContainers](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_08)

low-fragmentation heaps (LFH), [335](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_335)–[336](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_336)

### **M**

managing

power. *See* [power manager](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_45)

working sets, [417](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_417)–[421](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_421)

mandatory labels (SIDs), [630](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_630)

mapped page writer, [438](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_438)–[439](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_439)

mapped-file I/O, [513](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_513)

mappings (power), [595](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_595)

masks

access masks, [624](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_624)

affinity masks. *See* [affinity masks](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_06)

memory

address spaces. *See* [address spaces](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_04)

address translation. *See* [address translation](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_05)

AWE (Address Windowing Extensions), [22](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_22), [323](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_323)–[324](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_324)

combining, [459](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_459)–[467](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_467)

compression, [449](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_449)–[456](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_456)

Driver Verifier

IRQL checking, [557](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_557)

low resources simulation, [557](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_557)–[558](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_558)

miscellaneous checks, [558](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_558)–[559](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_559)

pool tracking, [556](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_556)–[557](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_557)

special pool, [555](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_555)–[556](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_556)

enclaves, [467](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_467)–[472](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_472)

heaps/pools. *See* [heaps/pools](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_22)

limits

physical memory, [446](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_446)–[447](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_447)

Windows clients, [447](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_447)–[449](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_449)

Memory Compression process, [91](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_91)

memory manager. *See* [memory manager](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_32)

NUMA (non-uniform memory architecture). *See* [NUMA](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_35)

page faults. *See* [page faults](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_36)

partitions, [456](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_456)–[458](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_458)

PFN. *See* [PFN](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_42)

section objects, [405](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_405)–[412](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_412)

stacks, [398](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_398)

SuperFetch. *See* [SuperFetch](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_63)

VADs (virtual address descriptors). *See* [VADs (virtual address descriptors)](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_71)

virtual memory, [21](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_21)–[23](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_23)

working sets

WSRM, [222](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_222)–[223](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_223)

Memory Compression process, [91](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_91)

memory manager

allocating memory, [310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)–[315](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_315)

attaching to the process, [310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)

AWE, [323](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_323)–[324](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_324)

commit charge, [313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

commit limit, [313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

committed pages, [310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)–[313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

components, [302](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_302)–[303](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_303)

copy-on-write, [321](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_321)–[323](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_323)

DEP, [319](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_319)–[321](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_321)

free pages, [310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)–[313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

granularity, [314](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_314)–[315](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_315)

internal synchronization, [308](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_308)

large pages, [303](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_303)–[304](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_304)

lazy evaluation, [323](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_323)

locking memory, [314](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_314)

memory-mapped files, [315](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_315)–[317](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_317)

NX page protection, [319](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_319)

overview, [301](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_301)–[302](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_302)

pages, defined, [304](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_304)

protecting memory, [317](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_317)–[319](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_319)

reserved pages, [310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)–[313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

services overview, [309](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_309)–[310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)

shareable pages, [310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)–[313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

shared memory, [315](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_315)

small pages, [303](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_303)–[304](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_304)

viewing memory, [305](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_305)–[308](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_308)

memory-mapped files, [315](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_315)–[317](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_317)

minimal processes, [104](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_104), [120](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_120)

minimum TCB list, [117](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_117)

miscellaneous checks (Device Driver), [558](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_558)–[559](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_559)

mitigating exploits. *See* [exploit mitigation](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_20)

mitigations (security), [370](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_370)

MMCSS (Multimedia Class Scheduler Service), [239](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_239), [251](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_251)–[254](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_254)

model (operating system), [46](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_46)–[47](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_47)

modern apps, [103](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_103)

modern processes, [103](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_103)–[104](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_104)

Modern Standby, [594](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_594)

modified page lists, [430](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_430)–[435](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_435)

modified page writer, [438](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_438)–[439](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_439)

modules (image loader), [164](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_164)–[168](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_168)

monitors (silos), [187](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_187)–[188](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_188)

MSV1\_0 authentication, [713](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_713)–[714](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_714)

Multimedia Class Scheduler Service (MMCSS), [239](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_239), [251](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_251)–[254](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_254)

multiple sessions, [29](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_29)–[30](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_30)

multiprocessor systems

affinity masks, [53](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_53), [275](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_275)–[276](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_276)

asymmetric, [51](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_51)

CPU sets, [278](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_278)–[283](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_283)

extended affinity masks, [276](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_276)–[277](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_277)

heterogenous, [52](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_52)

ideal node, [278](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_278)

ideal processor, [277](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_277)–[278](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_278)

last processor, [277](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_277)–[278](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_278)

NUMA systems, [269](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_269)–[271](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_271)

overview, [268](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_268)

processors

groups, [53](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_53), [271](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_271)–[273](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_273)

number per group, [273](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_273)

selecting, [284](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_284)–[286](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_286)

state, [274](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_274)

scheduler scalability, [274](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_274)

selecting, [283](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_283)–[284](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_284)

SMT systems, [268](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_268)–[269](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_269)

symmetric, [51](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_51)–[53](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_53)

multitasking (operating system), [51](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_51)

### **N**

namespaces, [703](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_703)–[705](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_705)

native images, [72](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_72)

native processes, [104](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_104)

.NET Framework, [6](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_6)–[7](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_7)

nodes

devnodes, [563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)–[569](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_569)

ideal node, [278](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_278)

processors, [52](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_52)

no-execute (NX) page protection, [319](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_319)

non-paged pools, [325](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_325)

non-uniform memory architecture. *See* [NUMA](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_35)

notification events, [423](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_423)–[425](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_425)

NT heaps, [334](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_334)

Ntdll.dll, [70](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_70)–[72](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_72)

NTOWF/TGT key, [613](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_613)–[614](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_614)

NUMA (non-uniform memory architecture)

overview, [404](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_404)

processors, [270](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_270)–[271](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_271)

systems, [269](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_269)–[271](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_271)

numbers

processors per group, [273](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_273)

threads, creating maximum, [399](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_399)

NX (no-execute) page protection, [319](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_319)

### **O**

objects

device drivers

device objects, [500](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_500)–[507](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_507)

file objects, [507](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_507)–[510](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_510)

driver objects

device drivers, [500](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_500)–[507](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_507)

dispatch routines, [504](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_504)

EPROCESS object, [138](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_138)–[140](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_140)

executive process object, [138](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_138)–[143](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_143)

file mapping objects, [20](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_20)

IoCompletion, [542](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_542)

jobs. *See* [jobs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_27)

kernel objects, [75](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_75)

namespaces (AppContainers), [703](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_703)–[705](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_705)

object access auditing, [679](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_679)–[681](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_681)

overview, [30](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_30)–[31](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_31)

section objects, [405](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_405)–[412](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_412)

security

access checks, [621](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_621)–[624](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_624)

ACEs. *See* [ACEs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_02)

ACLs. *See* [ACLs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_03)

DAC, [666](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_666)

overview, [619](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_619)–[621](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_621)

security descriptors, [650](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_650)–[656](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_656)

SIDs. *See* [SIDs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_58)

virtual service accounts, [646](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_646)–[650](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_650)

OneCore, [3](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_3)–[4](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_4)

opening

command prompt windows, [13](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_13)

devices, [507](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_507)–[510](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_510)

images, [135](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_135)–[138](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_138)

operating system (OS)

assertions, [753](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_753)

model, [46](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_46)–[47](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_47)

multitasking, [51](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_51)

scalability, [53](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_53)

OTS (over-the-shoulder ) elevation, [729](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_729)

output. *See* [I/O](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_23)

over-the-shoulder (OTS) elevation, [729](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_729)

Owner Rights SIDs, [662](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_662)

### **P**

packets. *See* [IRPs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_26)

PAE (Physical Address Extension), [371](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_371)

page directory entry (PDE), [374](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_374)

page directory pointer entry (PDPE), [374](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_374)

page directory pointer table (PDPT), [372](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_372)

page faults

clustered page faults, [387](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_387)–[388](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_388)

collided page faults, [387](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_387)

commit charge

commit limit, [394](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_394)–[396](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_396)

page file size, [397](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_397)–[398](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_398)

in-paging I/O, [386](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_386)–[387](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_387)

overview, [383](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_383)–[384](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_384)

page files

overview, [389](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_389)–[390](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_390)

swap files, [392](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_392)–[393](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_393)

viewing, [390](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_390)–[392](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_392)

virtual page files, [393](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_393)

PTEs

invalid PTEs, [384](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_384)–[385](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_385)

prototype PTEs, [385](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_385)–[386](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_386)

soft page faults, [384](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_384)

page files

overview, [389](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_389)–[390](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_390)

reservations (PFNs), [443](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_443)–[446](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_446)

size (commit charge), [397](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_397)–[398](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_398)

swap files, [392](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_392)–[393](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_393)

viewing, [390](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_390)–[392](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_392)

virtual page files, [393](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_393)

page frame number. *See* [PFN](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_42)

page lists, [428](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_428)–[435](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_435)

page table entries. *See* [PTEs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_50)

page tables, address translation, [375](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_375)–[376](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_376)

paged pools, [325](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_325)

pageheap, [343](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_343)–[346](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_346)

pages

combining

memory combining, [462](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_462)

releasing, [464](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_464)–[465](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_465)

committed pages, [310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)–[313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

defined, [304](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_304)

faults. *See* [page faults](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_36)

files. *See* [page files](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_37)

free pages, [310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)–[313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

mapped page writer, [438](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_438)–[439](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_439)

memory manager, [310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)–[313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

modified page writer, [438](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_438)–[439](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_439)

page frame number. *See* [PFN](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_42)

page lists. *See* [page lists](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_38)

page tables, address translation, [375](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_375)–[376](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_376)

paged pools, [325](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_325)

pageheap, [343](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_343)–[346](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_346)

PDE (page directory entry), [374](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_374)

PDPE (page directory pointer entry), [374](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_374)

PDPT (page directory pointer table), [372](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_372)

PFN. *See* [PFN](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_42)

priorities, [436](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_436)–[437](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_437), [476](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_476)–[478](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_478)

PTEs (page table entries). *See* [PTEs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_50)

reserved pages, [310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)–[313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

shareable pages, [310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)–[313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

states, [425](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_425)–[428](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_428)

SuperFetch, [476](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_476)–[478](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_478)

parameters, validating, [131](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_131)–[135](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_135)

parsing DLLs, [168](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_168)–[170](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_170)

partitions, [456](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_456)–[458](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_458)

passwords (Credential Guard), [613](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_613)

patches (kernel). *See* patches (kernels)

HyperGuard, [768](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_768)–[769](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_769)

overview, [764](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_764)–[765](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_765)

PatchGuard, [765](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_765)–[768](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_768)

PatchGuard, [765](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_765)–[768](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_768)

PCB (process control block), [106](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_106)

PCR (processor control region), [260](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_260)

PDE (page directory entry), [374](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_374)

PDPE (page directory pointer entry), [374](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_374)

PDPT (page directory pointer table), [372](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_372)

PEB (Process Environmental Block), [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105)

overview, [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105)

setting up, [143](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_143)

viewing, [110](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_110)–[111](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_111)

performance

Performance Monitor. *See* [Performance Monitor](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_41)

robust performance, [478](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_478)–[479](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_479)

states, [601](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_601)

SuperFetch, [478](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_478)–[479](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_479)

Performance Monitor

kernel mode/user mode comparison, [26](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_26)–[27](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_27)

overview, [36](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_36)–[38](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_38)

PFN (page frame number)

data structures, [440](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_440)–[443](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_443)

entries, [443](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_443)

mapped page writer, [438](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_438)–[439](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_439)

modified page writer, [438](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_438)–[439](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_439)

overview, [425](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_425)–[428](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_428)

page files (reservations), [443](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_443)–[446](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_446)

page lists, [428](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_428)–[435](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_435)

page priorities, [436](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_436)–[437](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_437)

page states, [425](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_425)–[428](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_428)

viewing, [427](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_427)–[428](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_428)

Physical Address Extension (PAE), [371](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_371)

physical memory limits, [446](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_446)–[447](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_447)

Pico

creating processes, [104](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_104)

overview, [121](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_121)–[122](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_122)

subsystems, [68](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_68)–[70](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_70)

placement policies (working sets), [416](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_416)–[417](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_417)

Plug and Play (PnP)

devices

catalog files, [574](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_574)

device stacks, [563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)–[569](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_569)

devnodes, [563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)–[569](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_569)

driver installation, [571](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_571)–[575](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_575)

driver support, [569](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_569)–[571](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_571)

enumeration, [561](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_561)–[563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)

INF files, [573](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_573)

support, [560](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_560)–[561](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_561)

trees, [561](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_561)–[563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)

drivers, [560](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_560)–[561](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_561)

overview, [559](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_559)–[560](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_560)

policies

advanced audit policy, [683](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_683)–[684](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_684)

authentication policies, [616](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_616)–[617](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_617)

Credential Guard, [616](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_616)–[617](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_617)

global audit policy, [682](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_682)–[683](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_683)

process mitigation policies, [735](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_735)–[740](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_740)

SRPs (software restriction policies), [757](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_757), [762](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_762)–[764](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_764)

trustlets, [124](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_124)–[125](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_125)

Windows edition enabled features, [56](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_56)–[57](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_57)

pools

size, [326](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_326)–[327](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_327)

threads, [299](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_299)–[300](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_300)

usage, [327](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_327)–[329](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_329)

port drivers, [494](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_494)–[495](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_495)

portability (Windows), [50](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_50)–[51](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_51)

ports

completion ports, [541](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_541)–[546](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_546)

port drivers, [494](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_494)–[495](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_495)

power manager

Connected Standy, [594](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_594)

controlling power, [599](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_599)–[600](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_600)

drivers, [596](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_596)

Legacy Standy, [594](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_594)

Modern Standy, [594](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_594)

overview, [590](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_590)–[594](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_594)

performance states, [601](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_601)

power availability requests, [602](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_602)–[603](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_603)

power management framework, [600](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_600)–[601](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_601)

power mapings, [595](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_595)

power states, [590](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_590)–[594](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_594)

system capabilities, [597](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_597)–[599](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_599)

PPL (Protected Processes Light), [115](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_115)–[120](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_120)

preemption (threads), [257](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_257)–[258](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_258)

prefetcher (working sets), [413](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_413)–[416](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_416)

prefixes (functions), [87](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_87)–[88](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_88)

priorities. *See also* [priority boosts](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_47)

I/O

bandwidth reservation, [551](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_551)–[552](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_552)

boosts, [549](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_549), [551](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_551)

bumps, [549](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_549), [551](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_551)

inversion, [549](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_549)

overview, [546](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_546)

strategies, [547](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_547)–[548](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_548)

viewing throughput, [549](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_549)–[551](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_551)

pages, [436](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_436)–[437](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_437)

SuperFetch, [476](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_476)–[478](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_478)

threads

levels, [215](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_215)–[219](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_219)

real-time, [218](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_218)–[219](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_219)

viewing, [219](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_219)–[222](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_222)

priority boosts. *See also* [priorities](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_46)

applying, [249](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_249)

Autoboost, [254](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_254)

balance set manager, [247](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_247)

CPU starvation, [246](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_246)–[248](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_248)

deadline scheduling, [254](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_254)

dispatch events, [239](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_239)–[240](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_240)

executive resources, [242](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_242)–[243](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_243)

foreground threads, [243](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_243)–[245](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_245)

games, [251](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_251)–[254](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_254)

GUI threads, [245](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_245)–[246](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_246)

I/O, [241](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_241)–[242](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_242)

locks, [241](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_241)

MMCSS, [239](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_239), [251](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_251)–[254](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_254)

multimedia, [251](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_251)–[254](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_254)

overview, [238](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_238)–[239](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_239)

priority inversion, [246](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_246)

removing, [250](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_250)

scheduling category, [251](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_251)

unwait boosts, [240](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_240)–[241](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_241)

private pages (committed pages), [310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)–[313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

privileges (accounts), [668](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_668)–[675](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_675)

Bypass Traverse Checking privilege, [675](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_675)

super privileges, [675](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_675)–[676](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_676)

.process, [109](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_109)–[110](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_110)

process control block (PCB), [106](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_106)

Process Environmental Block (PEB), [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105)

overview, [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105)

setting up, [143](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_143)

viewing, [110](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_110)–[111](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_111)

Process Explorer, [14](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_14)–[18](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_18)

process reflection (SuperFetch), [480](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_480)–[482](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_482)

process tree, viewing, [12](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_12)–[14](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_14)

process VADs, [402](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_402)–[403](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_403)

processes. *See also* [applications](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_09)

access tokens, [677](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_677)

attaching to the process, [310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)

console host, [67](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_67)

creating

address space, [140](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_140)–[142](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_142)

converting attributes, [131](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_131)–[135](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_135)

converting flags, [131](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_131)–[135](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_135)

executing initial thread, [148](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_148)

executive process object, [138](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_138)–[143](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_143)

initial thread, [144](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_144)–[146](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_146)

initializing process, [148](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_148)–[149](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_149)

initializing subsystem, [146](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_146)–[147](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_147)

kernel process structure, [141](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_141)

opening images, [135](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_135)–[138](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_138)

overview, [101](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_101)–[104](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_104), [129](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_129)–[130](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_130)

setting up EPROCESS object, [138](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_138)–[140](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_140)

setting up PEB, [143](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_143)

tracing startup, [149](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_149)–[154](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_154)

validating parameters, [131](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_131)–[135](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_135)

data structures

!process command, [109](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_109)

CSR\_PROCESS, [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105), [111](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_111)–[112](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_112)

DXGPROCESS, [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105)

EPROCESS, [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105)–[108](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_108)

ETHREAD, [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105)

KPROCESS, [106](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_106)–[107](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_107)

W32PROCESS, [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105), [113](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_113)

debugging unkillable processes, [539](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_539)–[541](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_541)

heaps, [333](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_333)

image loader. *See* [image loader](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_24)

immersive processes, [103](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_103)–[104](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_104)

jobs. *See* [jobs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_27)

kernel processes, [106](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_106)

minimal processes, [104](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_104), [120](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_120)

minimum TCB list, [117](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_117)

mitigation policies, [735](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_735)–[740](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_740)

modern processes, [103](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_103)–[104](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_104)

native processes, [104](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_104)

overview, [8](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_8)–[18](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_18)

PCB (process control block), [106](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_106)

PEB (Process Environmental Block). *See* [PEB (Process Environmental Block)](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_40)

Pico. *See* [Pico](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_43)

protected processes. *See* [protected processes](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_49)

reflection (SuperFetch), [480](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_480)–[482](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_482)

secure processes. *See* [trustlets](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_54)

system processes. *See* [system processes](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_64)

terminating, [154](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_154)–[155](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_155)

trustlets. *See* [trustlets](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_54)

UWP processes, [687](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_687)–[692](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_692)

VADs, [402](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_402)–[403](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_403)

viewing

DLL load search order, [163](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_163)–[164](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_164)

image loader, [156](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_156)–[157](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_157), [163](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_163)–[164](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_164)

PEB, [110](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_110)–[111](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_111)

Process Explorer, [14](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_14)–[18](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_18), [118](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_118)–[119](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_119)

process tree, [12](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_12)–[14](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_14)

protected processes, [118](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_118)–[119](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_119), [212](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_212)–[213](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_213)

Task Manager, [9](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_9)–[11](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_11)

processing I/O, [486](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_486)–[488](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_488)

processor control region (PCR), [260](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_260)

processors

groups

assigning, [271](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_271)–[273](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_273)

number of processors per group, [273](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_273)

scheduling (dynamic processors), [295](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_295)–[296](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_296)

symmetric multiprocessing, [53](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_53)

ideal processor, [277](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_277)–[278](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_278)

KPCR, [76](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_76)–[78](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_78)

KPRCB, [76](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_76)–[78](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_78)

last processor, [277](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_277)–[278](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_278)

nodes, [52](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_52)

multiprocessor systems. *See* multiprocessing/[multiprocessor systems](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_33)

NUMA, [270](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_270)–[271](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_271)

PCR (processor control region), [260](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_260)

selecting, [284](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_284)–[286](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_286)

SMT, [268](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_268)–[269](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_269)

state, [274](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_274)

programs. *See* [applications](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_09)

Protected Process Light (PPL), [115](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_115)–[120](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_120)

protected processes

overview, [113](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_113)–[115](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_115)

PPLs, [115](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_115)–[120](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_120)

viewing

Process Explorer, [118](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_118)–[119](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_119)

threads, [212](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_212)–[213](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_213)

protecting memory, [317](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_317)–[319](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_319)

prototype PTEs, [385](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_385)–[386](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_386)

PspInsertProcess function, [104](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_104), [143](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_143)

PTEs (page table entries)

address spaces, [355](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_355)–[356](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_356)

address translation, [375](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_375)–[376](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_376)

creating shared PTEs, [462](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_462)–[464](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_464)

defined, [372](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_372)

invalid PTEs, [384](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_384)–[385](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_385)

prototype PTEs, [385](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_385)–[386](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_386)

### **Q**

quantums (threads), [258](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_258)–[260](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_260)

accounting, [233](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_233)

clock cycles, [233](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_233)–[234](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_234)

clock interval, [232](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_232)

configuring, [237](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_237)–[238](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_238)

controlling, [234](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_234)–[235](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_235)

overview, [231](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_231)–[232](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_232)

registry value, [236](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_236)–[237](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_237)

variable quantums, [235](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_235)–[236](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_236)

quotas (address spaces), [364](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_364)–[365](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_365)

### **R**

randomization (user address spaces)

heap, [369](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_369)

images, [367](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_367)–[369](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_369)

stacks, [369](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_369)

viewing support, [370](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_370)–[371](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_371)

ratings (security)

CC, [607](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_607)

TCSEC, [605](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_605)–[607](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_607)

ready threads, [230](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_230)–[231](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_231)

ReadyBoost, [479](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_479)–[480](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_480)

ReadyBoot, [413](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_413)–[416](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_416)

ReadyDrive, [480](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_480)

real-time priorities (threads), [218](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_218)–[219](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_219)

reflection (process reflection), [480](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_480)–[482](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_482)

registry

overview, [32](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_32)–[33](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_33)

threads (quantum), [236](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_236)–[237](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_237)

viewing (security keys), [610](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_610)

virtualization (UAC), [722](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_722)–[724](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_724), [727](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_727)–[728](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_728)

releasing combined pages, [464](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_464)–[465](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_465)

removing priority boosts, [250](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_250)

requests. *See* [IRPs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_26)

reservations (PFNs), [443](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_443)–[446](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_446)

reserved pages (memory manager), [310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)–[313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

reserving bandwidth, [551](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_551)–[552](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_552)

Resource Monitor, [36](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_36)–[38](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_38)

resources

Driver Verifier, [557](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_557)–[558](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_558)

low resources simulation, [557](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_557)–[558](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_558)

Resource Monitor, [36](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_36)–[38](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_38)

restricted tokens (SIDs), [644](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_644)–[645](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_645)

rights (accounts), [668](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_668)–[670](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_670)

robust performance, [478](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_478)–[479](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_479)

rotate VADs, [403](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_403)

routines

device drivers, [498](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_498)–[500](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_500)

dispatch routines. *See* [dispatch routines](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_17)

overview, [7](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_7)–[8](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_8)

### **S**

SACLs. *See* [ACLs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_03)

safe DLL search mode, [160](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_160)

sandboxes (lowboxes), [134](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_134)

SAS implementation, [712](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_712)

saturation values, [216](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_216)

scalability

operating system, [53](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_53)

scheulers, [274](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_274)

scatter/gather I/O, [513](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_513)

scenarios (SuperFetch), [475](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_475)–[476](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_476)

schedulers (scalability), [274](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_274)

scheduling

groups

CPU rate limits, [292](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_292)–[295](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_295)

DFSS, [289](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_289)–[292](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_292)

dynamic processors, [295](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_295)–[296](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_296)

overview, [287](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_287)–[289](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_289)

priority boosts

deadline scheduling, [254](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_254)

scheduling category, [251](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_251)

schedulers (scalability), [274](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_274)

threads

context switches, [215](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_215)

dispatchers, [215](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_215)

exiting, [260](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_260)

heterogenous, [286](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_286)–[287](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_287)

preemption, [257](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_257)–[258](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_258)

quantums, [258](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_258)–[260](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_260)

terminating, [260](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_260)

voluntary switching, [256](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_256)–[257](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_257)

scheduling category, [251](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_251)

SDK (software development kit), [43](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_43)

searching

image loader, [160](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_160)

memory combining, [460](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_460)

safe DLL search mode, [160](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_160)

section objects, [405](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_405)–[412](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_412)

secure communication (Credential Guard), [614](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_614)–[615](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_615)

secure kernel, [59](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_59)–[61](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_61)

secure processes (trustlets)

attributes, [125](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_125)

built-in, [125](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_125)

identifying, [129](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_129)

identities, [125](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_125)–[126](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_126)

overview, [61](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_61), [123](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_123)–[124](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_124)

policies, [124](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_124)–[125](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_125)

services, [127](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_127)–[128](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_128)

system calls, [128](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_128)

Secure System process, [91](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_91)

security

access tokens. *See* [access tokens](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_01)

accounts

privileges, [668](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_668)–[675](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_675)

privileges, Bypass Traverse Checking privilege, [675](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_675)

privileges, super privileges, [675](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_675)–[676](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_676)

rights, [668](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_668)–[670](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_670)

AppContainers. *See* [AppContainers](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_08)

AppIDs, [756](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_756)–[757](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_757)

AppLocker, [757](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_757)–[762](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_762)

auditing. *See* [auditing (security)](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_11)

AuthZ API, [666](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_666)–[667](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_667)

CBAC, [667](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_667)

components, [608](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_608)–[611](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_611)

Credential Guard. *See* [Credential Guard](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_14)

Device Guard, [617](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_617)–[619](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_619)

exploit mitigation. *See* [exploit mitigation](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_20)

GUI security editors, [664](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_664)–[665](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_665)

heaps, [341](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_341)–[342](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_342)

IBAC, [667](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_667)

kernel patches. *See* [kernel patches](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_30)

logon. *See* [logon](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_31)

mitigations (user address spaces), [370](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_370)

objects

access checks, [621](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_621)–[624](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_624)

ACEs. *See* [ACEs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_02)

ACLs. *See* [ACLs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_03)

DAC, [666](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_666)

overview, [619](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_619)–[621](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_621)

security descriptors. *See* [security descriptors](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_55), [650](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_650)

SIDs. *See* [SIDs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_58), [625](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_625)

virtual service accounts, [646](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_646)–[650](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_650)

overview, [31](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_31)–[32](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_32), [605](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_605)

ratings. *See* [ratings (security)](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_53)

secure communication (Credential Guard), [614](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_614)–[615](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_615)

secure kernel, [59](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_59)–[61](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_61)

secure processes. *See* [secure processes (trustlets)](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_54)

secure system calls (functions), [71](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_71)

Secure System process, [91](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_91)

security descriptors. *See* [security descriptors](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_55)

SIDs (security identifiers). *See* [SIDs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_58)

SRPs, [757](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_757), [762](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_762)–[764](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_764)

trustlets. *See* [trustlets](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_54)

UAC (User Account Control). See UAC (User Account Control)

UIPI, [660](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_660)–[661](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_661)

VBS (virtualization-based security). See VBS (virtualization-based security)

architecture, [59](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_59)–[61](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_61)

hypervisor, [28](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_28)

viewing (registry keys), [610](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_610)

virtualization

files, [722](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_722)–[727](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_727)

overview, [611](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_611)–[612](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_612), [722](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_722)

registry, [722](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_722)–[724](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_724), [727](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_727)–[728](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_728)

VSM. *See* VSB

security descriptors

overview, [650](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_650)–[653](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_653)

viewing, [654](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_654)–[656](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_656)

security identifiers. *See* [SIDs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_58)

segment heaps, [336](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_336)–[337](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_337)

selecting

processors, [284](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_284)–[286](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_286)

threads, [266](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_266)–[267](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_267), [283](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_283)–[284](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_284)

server silos. *See* [silos](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_60)

Service Control Manager, [96](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_96)

service processes, [97](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_97)–[98](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_98)

services

memory manager, [309](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_309)–[310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)

overview, [7](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_7)–[8](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_8)

Service Control Manager, [96](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_96)

service processes, [97](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_97)–[98](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_98)

system service dispatcher, [70](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_70)–[71](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_71)

trustlets, [127](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_127)–[128](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_128)

viewing, [97](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_97)

Session Manager process, [92](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_92)–[95](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_95)

sessions

multiple, [29](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_29)–[30](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_30)

viewing, [353](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_353)–[355](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_355)

Session Manager process, [92](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_92)–[95](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_95)

x86 session space, [353](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_353)–[355](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_355)

setting

address limits, [363](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_363)–[364](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_364)

EPROCESS object, [138](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_138)–[140](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_140)

PEB, [143](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_143)

UAC, [733](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_733)–[734](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_734)

shareable pages, [310](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_310)–[313](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_313)

shared memory, [315](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_315)

sharing PTEs, [462](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_462)–[464](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_464)

showing. *See* [viewing](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_74)

SIDs (security identifiers)

filtered admin tokens, [645](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_645)–[646](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_646)

impersonation, [642](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_642)–[644](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_644)

integrity levels, [628](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_628)–[631](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_631), [641](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_641)–[642](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_642)

mandatory labels, [630](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_630)

overview, [625](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_625)–[626](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_626)

Owner Rights SIDs, [662](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_662)

restricted tokens, [644](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_644)–[645](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_645)

tokens, [632](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_632)–[640](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_640)

trust SIDs, [657](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_657)–[658](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_658)

viewing, [626](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_626)–[627](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_627)

silos (jobs)

ancillary functionality, [189](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_189)–[190](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_190)

containers, [190](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_190)–[191](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_191)

contexts, [186](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_186)–[188](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_188)

creating, [188](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_188)–[189](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_189)

isolation, [184](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_184)–[186](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_186)

monitors, [187](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_187)–[188](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_188)

objects, [183](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_183)–[184](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_184)

overview, [183](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_183)

simulating low resources, [557](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_557)–[558](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_558)

size

large address spaces, checking support, [351](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_351)

page files

commit charge, [397](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_397)–[398](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_398)

memory manager, [303](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_303)–[304](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_304)

pools, [325](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_325)–[327](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_327)

small pages (memory manager), [303](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_303)–[304](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_304)

SMT systems, [268](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_268)–[269](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_269)

soft page faults, [384](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_384)

software development kit (SDK), [43](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_43)

software PTEs, [384](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_384)–[385](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_385)

software restriction policies (SRPs), [757](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_757), [762](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_762)–[764](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_764)

special pools, [325](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_325), [555](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_555)–[556](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_556)

SRPs (software restriction policies), [757](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_757), [762](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_762)–[764](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_764)

stacks

device stacks, [563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)–[569](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_569)

DPC stacks, [401](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_401)

IRPs, [515](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_515)–[519](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_519), [521](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_521)

kernel stacks, [400](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_400)–[401](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_401)

overview, [398](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_398)

Plug and Play, [563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)–[569](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_569)

randomizing, [369](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_369)

user address spaces, [369](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_369)

user stacks, [399](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_399)

standby

SuperFetch, [475](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_475)

page lists, [430](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_430)–[435](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_435)

startup (subsystems), [63](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_63)–[64](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_64)

states

pages (PFN), [425](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_425)–[428](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_428)

processors, [274](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_274)

threads, [223](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_223)–[228](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_228)

storage (TLS), [18](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_18)

strategies (I/O priorities), [547](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_547)–[548](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_548)

strengthening CFG, [747](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_747)–[749](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_749)

structures

data structures. *See* [data structures](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_15)

kernel process structure, [141](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_141)

subsystems

console host, [67](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_67)

DLLs, [62](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_62)–[63](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_63)

initializing, [146](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_146)–[147](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_147)

Linux, [68](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_68)–[70](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_70)

native images, [72](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_72)

Ntdll.dll, [70](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_70)–[72](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_72)

overview, [62](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_62)–[63](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_63)

Pico providers, [68](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_68)–[70](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_70)

startup, [63](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_63)–[64](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_64)

subsystem DLLs, [48](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_48)

viewing types, [62](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_62)–[63](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_63)

Windows subsystem, [64](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_64)–[67](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_67)

super privileges, [675](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_675)–[676](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_676)

SuperFetch

components, [473](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_473)–[474](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_474)

fast user switching, [475](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_475)

hibernation, [475](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_475)

logging, [474](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_474)–[475](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_475)

overview, [472](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_472)–[474](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_474)

page priorities, [476](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_476)–[478](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_478)

process reflection, [480](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_480)–[482](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_482)

ReadyBoost, [479](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_479)–[480](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_480)

ReadyDrive, [480](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_480)

robust performance, [478](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_478)–[479](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_479)

scenarios, [475](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_475)–[476](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_476)

standby, [475](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_475)

tracing, [474](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_474)–[475](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_475)

support (Plug and Play), [560](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_560)–[561](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_561), [569](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_569)–[571](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_571)

suppression (CFG), [741](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_741), [748](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_748)

suspending threads, [264](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_264)

swap files, [392](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_392)–[393](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_393)

swapper (working sets), [421](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_421)–[422](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_422)

switches/switching

context switches/switching, [215](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_215), [255](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_255)–[256](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_256)

Direct Switch, [255](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_255)–[256](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_256)

directed context switch, [19](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_19)

voluntary switching, [256](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_256)–[257](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_257)

symbols

kernel debugging, [38](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_38)

viewing kernel type information, [41](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_41)–[42](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_42)

symmetric multiprocessing, [51](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_51)–[53](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_53)

synchronization

heaps, [334](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_334)–[335](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_335)

internal synchronization, [308](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_308)

IRPs, [531](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_531)–[533](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_533)

memory, [308](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_308)

synchronous I/O, [511](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_511)

Sysinternals, [44](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_44)

system address space layouts (x86), [352](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_352)–[353](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_353)

System and Compressed Memory process, [90](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_90)

system calls (trustlets), [128](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_128)

system capabilities (power manager), [597](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_597)–[599](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_599)

System process, [90](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_90)–[91](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_91)

system processes

Idle process, [89](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_89)–[90](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_90)

Memory Compression process, [91](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_91)

overview, [88](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_88)–[89](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_89)

Secure System process, [91](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_91)

Service Control Manager, [96](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_96)

service processes, [97](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_97)–[98](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_98)

Session Manager process, [92](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_92)–[95](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_95)

System and Compressed Memory process, [90](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_90)

System process, [90](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_90)–[91](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_91)

system thread, [90](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_90)–[91](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_91)

Windows Initialization process, [95](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_95)–[96](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_96)

Windows logon process, [98](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_98)–[99](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_99)

system PTEs (page table entries)

address spaces, [355](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_355)–[356](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_356)

address translation, [375](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_375)–[376](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_376)

creating shared PTEs, [462](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_462)–[464](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_464)

defined, [372](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_372)

invalid PTEs, [384](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_384)–[385](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_385)

prototype PTEs, [385](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_385)–[386](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_386)

system services (functions), [72](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_72)

system thread, [90](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_90)–[91](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_91)

system working sets, [422](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_422)–[423](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_423)

systems

PTEs (page table entries). *See* PTEs (page table entries)

subsystems. *See* [subsystems](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_62)

Sysinternals, [44](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_44)

system address space layouts (x86), [352](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_352)–[353](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_353)

System and Compressed Memory process, [90](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_90)

system calls (trustlets), [128](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_128)

system capabilities (power manager), [597](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_597)–[599](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_599)

System process, [90](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_90)–[91](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_91)

system processes. *See* [system processes](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_64)

system services (functions), [72](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_72)

system thread, [90](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_90)–[91](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_91)

system working sets, [422](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_422)–[423](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_423)

SystemParametersInfo function, [178](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_178)

viewing (system service dispatcher), [70](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_70)–[71](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_71)

### **T**

tables (PTEs, page table entries)

address spaces, [355](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_355)–[356](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_356)

address translation, [375](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_375)–[376](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_376)

creating shared PTEs, [462](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_462)–[464](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_464)

defined, [372](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_372)

invalid PTEs, [384](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_384)–[385](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_385)

prototype PTEs, [385](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_385)–[386](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_386)

Task Manager, [9](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_9)–[11](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_11)

TCB (thread control block)

minimum TCB list, [117](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_117)

overview, [196](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_196)

TCSEC (Trusted Computer System Evaluation Criteria), [605](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_605)–[607](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_607)

TEB (thread environment block)

overview, [194](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_194), [198](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_198)

viewing, [201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)–[205](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_205)

Terminal Services, [29](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_29)–[30](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_30)

terminating

I/O, [539](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_539)

processes, [154](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_154)–[155](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_155)

TerminateJobObject function, [179](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_179)

threads, [260](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_260), [539](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_539)

thread-agnostic I/O, [536](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_536)–[537](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_537)

thread control block (TCB)

minimum TCB list, [117](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_117)

overview, [196](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_196)

thread environment block (TEB)

overview, [194](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_194), [198](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_198)

viewing, [201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)–[205](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_205)

Thread Information Block (TIB), [201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)

thread local storage (TLS), [18](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_18)

thread pools, [297](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_297)–[300](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_300)

threads

access tokens, [20](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_20), [677](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_677)

cancelling I/O, [539](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_539)

concurrency, [542](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_542)

context, [18](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_18)

context switching, [255](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_255)–[256](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_256)

contexts, [18](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_18)

cores, [52](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_52)

creating, [193](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_193)–[194](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_194), [206](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_206)–[207](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_207), [399](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_399)

data structures

CSR\_THREAD, [195](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_195), [205](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_205)–[206](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_206)

ETHREAD, [194](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_194)–[201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)

KTHREAD, [194](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_194)–[201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)

Direct Switch, [255](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_255)–[256](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_256)

directed context switch, [19](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_19)

dispatcher database, [228](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_228)–[230](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_230)

fibers, [19](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_19)

file mapping objects, [20](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_20)

freezing, [264](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_264)–[266](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_266)

group scheduling

CPU rate limits, [292](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_292)–[295](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_295)

DFSS, [289](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_289)–[292](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_292)

dynamic processors, [295](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_295)–[296](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_296)

overview, [287](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_287)–[289](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_289)

idle threads, [260](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_260)–[263](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_263), [267](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_267)

initial thread, [144](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_144), [145](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_145)–[146](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_146), [148](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_148)

maximum number, [399](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_399)

multiprocessor systems. *See* [multiprocessor systems](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_33)

overview, [18](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_18)–[19](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_19)

PCR, [260](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_260)

priorities

levels, [215](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_215)–[219](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_219)

real-time, [218](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_218)–[219](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_219)

viewing, [219](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_219)–[222](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_222)

priority boosts

applying, [249](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_249)

Autoboost, [254](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_254)

balance set manager, [247](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_247)

CPU starvation, [246](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_246)–[248](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_248)

deadline scheduling, [254](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_254)

dispatch events, [239](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_239)–[240](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_240)

executive resources, [242](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_242)–[243](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_243)

foreground threads, [243](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_243)–[245](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_245)

games, [251](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_251)–[254](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_254)

GUI threads, [245](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_245)–[246](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_246)

I/O, [241](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_241)–[242](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_242)

locks, [241](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_241)

MMCSS, [239](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_239), [251](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_251)–[254](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_254)

multimedia, [251](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_251)–[254](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_254)

overview, [238](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_238)–[239](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_239)

priority inversion, [246](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_246)

removing, [250](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_250)

scheduling category, [251](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_251)

unwait boosts, [240](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_240)–[241](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_241)

quantums. *See* [quantums (threads)](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_52)

ready, [230](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_230)–[231](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_231)

saturation values, [216](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_216)

scheduling

context switches, [215](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_215)

dispatchers, [215](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_215)

exiting, [260](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_260)

heterogenous, [286](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_286)–[287](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_287)

preemption, [257](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_257)–[258](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_258)

quantums, [258](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_258)–[260](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_260)

terminating, [260](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_260)

voluntary switching, [256](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_256)–[257](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_257)

scheduling overview, [214](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_214)–[215](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_215)

selecting, [266](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_266)–[267](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_267), [283](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_283)–[284](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_284)

states, [223](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_223)–[228](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_228)

suspending, [264](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_264)

system thread, [90](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_90)–[91](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_91)

TCB (thread control block). *See* [TCB (thread control block)](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_65)

TEB (thread environment block). *See* [TEB (thread environment block)](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_66)

terminating, [539](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_539)

thread-agnostic I/O, [536](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_536)–[537](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_537)

thread pools, [297](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_297)–[300](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_300)

TIB (Thread Information Block), [201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)

TLS (thread local storage), [18](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_18)

UMS threads, [19](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_19)–[20](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_20)

VADs (virtual address descriptors), [20](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_20)

viewing

kernel-mode debugger, [210](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_210)–[212](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_212)

overview, [207](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_207)–[209](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_209)

protected processes, [212](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_212)–[213](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_213)

ready, [230](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_230)–[231](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_231)

TEB, [201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)–[205](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_205)

user-mode debugger, [209](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_209)–[210](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_210)

worker factories, [297](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_297)–[300](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_300)

throughput (I/O priorities), [549](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_549)–[551](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_551)

TIB (Thread Information Block), [201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)

TLB (translation look-aside buffer), [377](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_377)–[378](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_378)

TLS (thread local storage), [18](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_18)

tokens

access tokens, [677](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_677)

AppContainters, [690](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_690)–[692](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_692)

BNO isolation, [708](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_708)

SIDs, [632](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_632)–[640](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_640)

stored handles, [706](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_706)–[708](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_708)

tools

Debugging Tools for Windows (kernel debugging), [38](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_38)–[42](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_42)

Windows, viewing internals, [35](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_35)–[36](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_36)

tracing

process startup, [149](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_149)–[154](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_154)

SuperFetch, [474](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_474)–[475](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_475)

tracking pools (Device Driver), [556](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_556)–[557](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_557)

translating addresses

ARM virtual address translation, [381](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_381)–[382](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_382)

overview, [371](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_371)

page tables, [375](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_375)–[376](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_376)

PTEs, [375](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_375)–[376](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_376)

TLB, [377](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_377)–[378](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_378)

viewing, [378](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_378)–[380](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_380)

write bits, [376](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_376)–[377](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_377)

x64 virtual address translation, [380](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_380)–[381](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_381)

x86 virtual address translation, [371](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_371)–[375](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_375)

translation look-aside buffer (TLB), [377](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_377)–[378](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_378)

trees (Plug and Play), [561](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_561)–[563](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_563)

troubleshooting pools, [329](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_329)–[330](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_330)

trust SIDs, [657](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_657)–[658](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_658)

Trusted Computer System Evaluation Criteria (TCSEC), [605](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_605)–[607](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_607)

trustlets

attributes, [125](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_125)

built-in, [125](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_125)

identifying, [129](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_129)

identities, [125](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_125)–[126](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_126)

overview, [61](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_61), [123](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_123)–[124](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_124)

policies, [124](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_124)–[125](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_125)

services, [127](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_127)–[128](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_128)

system calls, [128](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_128)

types

device drivers, [492](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_492)–[496](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_496)

heaps, [334](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_334)

kernels, [41](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_41)–[42](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_42)

subsystems, [62](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_62)–[63](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_63)

### **U**

UAC (User Account Control)

elevation

Admin Approval Mode (AAM), [729](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_729)

administrative rights, [729](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_729)–[732](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_732)

auto-elevation, [732](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_732)–[733](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_733)

consent, [729](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_729)

over-the-shoulder (OTS), [729](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_729)

overview, [729](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_729)

settings, [733](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_733)–[734](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_734)

overview, [722](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_722)

virtualization

files, [722](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_722)–[727](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_727)

registry, [722](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_722)–[724](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_724), [727](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_727)–[728](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_728)

UEFI (Credential Guard), [616](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_616)

UIPI (User Interface Privilege Isolation), [660](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_660)–[661](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_661)

UMDF (WDF), [578](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_578), [580](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_580)–[581](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_581), [587](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_587)–[590](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_590)

UMS threads (user mode scheduling threads), [19](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_19)–[20](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_20)

Unicode, [33](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_33)–[35](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_35)

Universal Windows Drivers, [85](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_85)

unkillable processes, debugging, [539](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_539)–[541](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_541)

unwait boosts, [240](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_240)–[241](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_241)

updating Windows, [3](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_3)

usage (pools), [327](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_327)–[329](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_329)

User Account Control. *See* [UAC](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_68)

user address spaces

calculating load address, [368](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_368)–[369](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_369)

EMET, [370](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_370)

heap randomization, [369](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_369)

image randomization, [367](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_367)–[369](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_369)

IRPs, [528](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_528)–[531](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_531)

kernel, [369](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_369)

layouts, [365](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_365)–[367](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_367)

overview, [365](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_365)–[367](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_367)

security mitigations, [370](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_370)

stack randomization, [369](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_369)

viewing, [366](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_366)–[367](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_367)

viewing randomization support, [370](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_370)–[371](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_371)

User Interface Privilege Isolation (UIPI), [660](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_660)–[661](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_661)

user mode

architecture, [47](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_47)–[49](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_49)

kernel mode comparison, [23](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_23)–[27](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_27), [46](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_46)

user-mode debugging

Debugging Tools for Windows, [39](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_39)

viewing threads, [209](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_209)–[210](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_210)

user mode scheduling threads (UMS threads), [19](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_19)–[20](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_20)

user stacks, [399](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_399)

users

authentication, [713](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_713)–[718](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_718)

Kerberos, [714](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_714)–[715](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_715)

MSV1\_0, [713](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_713)–[714](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_714)

viewing active logon sessions, [715](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_715)–[717](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_717)

cancelling I/O, [537](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_537)–[538](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_538)

fast user switching, [30](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_30), [475](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_475)

multiple sessions, [29](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_29)–[30](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_30)

SuperFetch, [475](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_475)

UAC. *See* [UAC](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_68)

user address spaces. *See* [user address spaces](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_70)

using virtual service accounts, [647](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_647)–[650](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_650)

UWP apps, [685](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_685)–[687](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_687)

UWP processes, [687](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_687)–[692](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_692)

### **V**

VADs (virtual address descriptors)

overview, [20](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_20), [401](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_401)

process VADs, [402](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_402)–[403](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_403)

rotate VADs, [403](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_403)

validating parameters (processes), [131](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_131)–[135](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_135)

values (saturation values), [216](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_216)

variable quantums, [235](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_235)–[236](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_236)

VBS (virtualization-based security)

architecture, [59](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_59)–[61](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_61)

hypervisor, [28](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_28)

versions (Windows)

OneCore, [3](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_3)–[4](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_4)

overview, [1](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_1)–[3](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_3)

updating, [3](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_3)

viewing

access masks, [624](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_624)

active logon sessions, [715](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_715)–[717](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_717)

addresses

randomization support, [370](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_370)–[371](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_371)

translation, [378](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_378)–[380](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_380)

usage, [361](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_361)–[363](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_363)

user address spaces, [366](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_366)–[367](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_367), [370](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_370)–[371](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_371)

VADs (virtual address descriptors), [402](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_402)–[403](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_403)

AppContainer

atom table, [697](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_697)–[698](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_698)

capabilities, [701](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_701)–[703](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_703)

security attributes, [695](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_695)–[697](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_697)

tokens, [690](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_690)–[692](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_692)

brokers, [709](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_709)

CFG, [742](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_742)–[744](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_744)

CFG bitmaps, [746](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_746)–[747](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_747)

control areas, [408](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_408)–[412](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_412)

CPU. *See* [CPU](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_13)

CSR\_PROCESS data structure, [112](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_112)

CSR\_THREAD structure, [206](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_206)

DEP, [321](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_321)

device drivers, [85](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_85)–[88](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_88)

device objects, [502](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_502)–[506](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_506)

devnodes, [568](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_568)–[569](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_569)

DFSS (dynamic fair share scheduling), [290](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_290)–[292](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_292)

drivers, [496](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_496)–[498](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_498)

catalog files, [574](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_574)

device drivers, [85](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_85)–[88](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_88)

dispatch routines, [517](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_517)–[518](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_518)

INF files, [573](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_573)

KMDF drivers, [580](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_580)–[581](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_581)

objects, [504](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_504)–[506](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_506)

power mappings, [596](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_596)

UMDF drivers, [580](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_580)–[581](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_581)

dumping. *See* [dumping](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_19)

enabling privileges, [673](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_673)–[675](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_675)

EPROCESS data structure, [107](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_107)–[108](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_108)

ETHREAD structure, [196](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_196)–[201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)

files

catalog files, [574](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_574)

file objects, [508](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_508)–[509](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_509)

INF files, [573](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_573)

memory-mapped files, [316](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_316)–[317](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_317)

page files, [390](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_390)–[392](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_392), [397](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_397)–[398](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_398)

prefetch file reads and writes, [415](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_415)–[416](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_416)

swap files, [393](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_393)

virtual page files, [393](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_393)

virtualization, [726](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_726)–[727](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_727)

free page lists, [429](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_429)–[430](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_430)

functions, exported, [34](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_34)–[35](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_35)

global audit policy, [682](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_682)

HAL image dependencies, [80](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_80)–[82](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_82)

heaps, [338](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_338)–[341](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_341)

image loader, [156](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_156)–[157](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_157)

DLL load search order, [163](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_163)–[164](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_164)

loaded modules database, [166](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_166)–[167](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_167)

integrity levels, [628](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_628)–[631](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_631)

I/O

fast I/O, [512](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_512)–[513](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_513)

IRPs, [518](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_518)–[519](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_519), [521](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_521)–[524](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_524)

priority boosting/bumping, [551](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_551)

priority throughput, [549](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_549)–[551](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_551)

IRPs, [518](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_518)–[519](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_519), [521](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_521)–[524](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_524)

jobs, [180](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_180)–[183](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03a.html#page_183)

kernel

kernel-mode debugger, [210](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_210)–[212](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_212)

KPRCB (kernel processor control block), [77](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_77)–[78](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_78)

KPCR (kernel processor control region), [77](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_77)–[78](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_78)

stacks, [400](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_400)–[401](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_401)

type information, [41](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_41)–[42](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_42)

KTHREAD structure, [196](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_196)–[201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)

look-aside lists, [331](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_331)–[332](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_332)

memory, [305](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_305)–[308](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_308)

combining, [465](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_465)–[467](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_467)

compression, [455](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_455)–[456](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_456)

memory-mapped files, [316](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_316)–[317](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_317)

partitions, [458](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_458)

modified page lists, [430](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_430)–[435](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_435)

notification events, [424](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_424)–[425](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_425)

NUMA processors, [270](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_270)–[271](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_271)

object access auditing, [679](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_679)–[681](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_681)

pages

files, [390](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_390)–[392](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_392), [397](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_397)–[398](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_398)

free page lists, [429](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_429)–[430](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_430)

modified page lists, [430](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_430)–[435](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_435)

pageheap, [344](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_344)–[346](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_346)

PFN (Page Frame Number), [427](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_427)–[428](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_428), [443](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_443)

priorities, [437](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_437)

PTEs (page table entries), [355](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_355)–[356](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_356)

standby page lists, [430](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_430)–[435](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_435)

virtual page files, [393](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_393)

zero page lists, [429](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_429)–[430](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_430)

PEB (Process Environmental Block), [110](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_110)–[111](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_111)

PFN (Page Frame Number), [427](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_427)–[428](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_428), [443](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_443)

pools. *See* [pools](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_22)

power

availability requests, [603](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_603)

driver power mappings, [596](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_596)

states, [592](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_592)–[593](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_593)

system capabilities, [597](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_597)–[599](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_599)

prefetch file reads and writes, [415](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_415)–[416](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_416)

priority boosts

bumping (I/O), [551](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_551)

CPU starvation, [247](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_247)–[248](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_248)

foreground threads, [243](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_243)–[245](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_245)

GUI threads, [245](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_245)–[246](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_246)

MMCSS (Multimedia Class Scheduler Service), [252](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_252)–[253](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_253)

processes

affinity, [275](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_275)–[276](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_276)

data structures, [109](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_109)

PEB (Process Environmental Block), [110](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_110)–[111](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_111)

Process Explorer, [14](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_14)–[18](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_18), [118](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_118)–[119](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_119)

process tree, [12](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_12)–[14](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_14)

protected processes, [118](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_118)–[119](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_119), [212](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_212)–[213](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_213)

service processes, [97](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_97)–[98](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_98)

Task Manager, [9](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_9)–[11](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_11)

UWP processes, [689](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_689)–[690](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_690)

PTEs (page table entries), [355](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_355)–[356](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_356)

section objects, [406](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_406)–[407](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_407)

security

AppContainer, [695](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_695)–[697](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_697)

descriptors, [654](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_654)–[656](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_656)

registry keys, [610](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_610)

SIDs (security identifiers), [626](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_626)–[627](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_627)

trust SIDs, [658](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_658)

services, [97](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_97)

MMCSS (Multimedia Class Scheduler Service), [252](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_252)–[253](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_253)

service processes, [97](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_97)–[98](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_98)

system service dispatcher, [70](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_70)–[71](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_71)

sessions, [353](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_353)–[355](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_355), [715](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_715)–[717](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_717)

SMT processors, [268](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_268)–[269](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_269)

SRPs (software restriction policies), [764](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_764)

stacks, [400](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_400)–[401](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_401), [521](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_521)

standby page lists, [430](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_430)–[435](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_435)

swap files, [393](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_393)

systems

power capabilities, [597](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_597)–[599](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_599)

subsystems, [62](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_62)–[63](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_63)

system service dispatcher, [70](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_70)–[71](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_71)

threads

clock cycles per quantum, [233](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_233)–[234](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_234)

clock interval, [232](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_232)

foreground threads, [243](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_243)–[245](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_245)

freezing, [265](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_265)–[266](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_266)

GUI threads, [245](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_245)–[246](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_246)

idle threads, [260](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_260)–[262](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_262)

kernel-mode debugger, [210](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_210)–[212](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_212)

KTHREAD structure, [196](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_196)–[201](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_201)

overview, [207](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_207)–[209](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_209)

pools, [299](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_299)–[300](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_300)

priorities, [219](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_219)–[222](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_222)

protected processes, [118](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_118)–[119](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_119), [212](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_212)–[213](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_213)

ready, [230](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_230)–[231](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_231)

states, [224](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_224)–[228](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_228)

user-mode debugger, [209](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_209)–[210](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_210)

tokens, [635](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_635)–[640](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_640)

AppContainers, [690](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_690)–[692](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_692)

filtered admin tokens, [645](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_645)–[646](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_646)

token stored handles, [706](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_706)–[708](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_708)

virtual page files, [393](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_393)

Windows

enabled features, [56](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_56)–[57](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_57)

tools, [35](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_35)–[36](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_36)

working sets, [418](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_418)–[421](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_421)

zero page lists, [429](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_429)–[430](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_430)

virtual address descriptors (VADs). *See* [VADs (virtual address descriptors)](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_71)

virtual address spaces. *See* [address spaces](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_04)

virtual memory, [21](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_21)–[23](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_23)

virtual page files, [393](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_393)

Virtual Secure Mode. *See* [VBS](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_72)

virtual service accounts, [646](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_646)–[650](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_650)

Virtual Trust Levels (VTLs), [59](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_59)–[61](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_61)

virtualization

security

authentication policies, [616](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_616)–[617](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_617)

Device Guard, [617](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_617)–[619](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_619)

files, [722](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_722)–[727](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_727)

Kerberos armoring, [616](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_616)–[617](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_617)

NTOWF/TGT key, [613](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_613)–[614](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_614)

overview, [611](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_611)–[613](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_613), [722](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_722)

passwords, [613](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_613)

registry, [722](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_722)–[724](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_724), [727](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_727)–[728](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_728)

secure communication, [614](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_614)–[615](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_615)

UEFI, [616](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07.html#page_616)

VBS (virtualization-based security). *See* VBS (virtualization-based security)

virtualization-based security (VBS): See VBS (virtualization-based security)

voluntary switching, [256](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_256)–[257](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_257)

VSM. *See* [VBS](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_72)

VTLs (Virtual Trust Levels), [59](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_59)–[61](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_61)

### **W**

W32PROCESS data structure, [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105), [113](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_113)

WBF (Windows Biometric Framework), [719](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_719)–[721](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_721)

WDF (Windows Driver Foundation)

KMDF, [578](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_578)–[587](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_587)

overview, [84](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_84), [578](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_578)–[579](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_579)

UMDF, [578](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_578), [580](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_580)–[581](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_581), [587](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_587)–[590](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_590)

WDK (Windows Driver Kit), [43](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_43)–[44](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_44)

WDM (Windows Driver Model), [83](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_83)–[84](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_84), [493](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_493)–[494](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_494)

windows, opening command prompts, [13](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_13)

Windows

client memory limits, [447](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_447)–[449](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_449)

design goals, [45](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_45)–[46](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_46)

editions

comparison, [54](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_54)–[56](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_56)

viewing enabled features, [56](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_56)–[57](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_57)

operating system model, [46](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_46)–[47](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_47)

portability, [50](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_50)–[51](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_51)

SDK (software development kit), [43](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_43)

versions

OneCore, [3](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_3)–[4](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_4)

overview, [1](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_1)–[3](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_3)

updating, [3](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_3)

viewing internals, [35](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_35)–[36](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_36)

W32PROCESS data structure, [105](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_105), [113](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch03.html#page_113)

WBF (Windows Biometric Framework), [719](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_719)–[721](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_721)

WDF (Windows Driver Foundation). *See* WDF Windows Driver Foundation)

WDK (Windows Driver Kit), [43](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_43)–[44](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_44)

WDM (Windows Driver Model), [83](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_83)–[84](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_84), [493](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_493)–[494](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_494)

Windows API. *See* [APIs](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/index.html#index_07)

Windows Hello, [721](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_721)

Windows Initialization process, [95](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_95)–[96](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_96)

Windows logon process, [98](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_98)–[99](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_99)

Windows Runtime, [5](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_5)–[6](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_6)

Windows subsystem, [64](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_64)–[67](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_67)

Winlogon initialization, [711](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_711)–[712](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_712)

winver command, [3](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_3)

WSRM (Windows System Resource Manager ), [222](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_222)–[223](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_223)

Windows Biometric Framework (WBF), [719](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_719)–[721](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_721)

Windows Driver Foundation (WDF)

KMDF, [578](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_578)–[587](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_587)

overview, [84](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_84), [578](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_578)–[579](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_579)

UMDF, [578](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_578), [580](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_580)–[581](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_581), [587](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_587)–[590](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06a.html#page_590)

Windows Driver Kit (WDK), [43](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_43)–[44](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_44)

Windows Driver Model (WDM), [83](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_83)–[84](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_84), [493](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_493)–[494](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch06.html#page_494)

Windows Hello, [721](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_721)

Windows Initialization process, [95](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_95)–[96](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_96)

Windows logon process, [98](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_98)–[99](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_99)

Windows Runtime, [5](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_5)–[6](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch01.html#page_6)

Windows System Resource Manager (WSRM), [222](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_222)–[223](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_223)

Winlogon initialization, [711](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_711)–[712](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch07a.html#page_712)

worker factories, [297](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_297)–[300](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04a.html#page_300)

working sets

balance set manager, [421](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_421)–[422](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_422)

demand paging, [413](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_413)

managing, [417](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_417)–[421](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_421)

memory, [412](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_412)

notification events, [423](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_423)–[425](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_425)

overview, [413](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_413)

placement policies, [416](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_416)–[417](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_417)

prefetcher, [413](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_413)–[416](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_416)

ReadyBoot, [413](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_413)–[416](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_416)

swapper, [421](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_421)–[422](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_422)

system working sets, [422](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_422)–[423](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_423)

viewing, [418](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_418)–[421](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_421)

write bits, [376](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_376)–[377](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_377)

WSRM (Windows System Resource Manager), [222](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_222)–[223](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch04.html#page_223)

### **X–Z**

x64 systems, [50](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch02.html#page_50)

x64 virtual address limitations, [359](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_359)

x64 virtual address translation, [380](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_380)–[381](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_381)

x86 address space layouts, [349](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_349)–[352](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_352)

x86 session space, [353](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_353)–[355](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_355)

x86 system address space layouts, [352](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_352)–[353](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05.html#page_353)

x86 virtual address translation, [371](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_371)–[375](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05a.html#page_375)

zero page lists, [429](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_429)–[430](https://learning.oreilly.com/library/view/windows-internals-seventh/9780133986471/ch05b.html#page_430)