WINDOWS 10 TRICKS & FIXES

Summary

Preface

This is not a typical eBook, It's a "rapid notes collection document" created to show synthetically procedures and information to fix generic problems that could happen with Windows and give useful tricks and shortcuts that may improve the Windows user experience.
I assume who read this eBook is an intermediate user that knows the basic Windows usage and the meaning of words like "*drivers*", "*command prompt*" "scandisk" "*defragmentation*" and so on. The target OS is Windows 10 although the largest part of contents is applicable to others previous Microsoft Windows OS.

This is only a preliminary draft, if you have suggestions feel free to contact me using the form on my website.

Shortcut Table

Keys	Command
Windows Key + Left	Snap current window to the left side of the screen.
Windows Key + Right	Snap current window the the right side of the screen.
Windows Key + Up	Snap current window to the top of the screen.
Windows Key + Down	Snap current window to the bottom of the screen.
Windows Key + Tab	Opens the Task View interface for current virtual desktop.
Alt + Tab	Switch between your open Windows on all virtual desktops.
Windows Key + Ctrl + D	Create a new virtual desktop and switch to it
Windows Key + Ctrl + F4	Close the current virtual desktop.
Windows Key + Ctrl + Left / Right	Switch to the virtual desktop on the left or right.
Windows key + C:	Opens Cortana in listening mode.
Windows key + I:	Access the Settings
Windows key + H	Access the Share charm
Windows key + K	Access the Devices charm
Windows key + Q	Opens Cortana
Windows key + P	Access the Second Screen mode bar
Windows key + X	Access the Windows Tools Menu
Windows key + +	Magnifying Zoom
Windows key + PrtScn	Takes a screenshot of the screen and automatically saves it in the Pictures folder and copy in clipboard
PrtScn	Copy in clipboard the screenshot of the screen
Windows key + Enter	Launch Narrator
Windows key + E	Show the most frequently accessed files and folders
Windows key + R	Open the Run dialog box
Windows key + U	Open Ease of Access Center
Windows key + Ctrl + F	Open Find Computers dialog box
Windows key + Pause/Break	Open the System page
Windows key + 1..10	Launch a program pinned on the Taskbar in the position indicated by the number
Windows key + Ctrl + 1..10	Access the last active instance of a program pinned on the Taskbar in the position indicated by the number
Windows key + Shift + 1..10	Launch a new instance of a program pinned on the Taskbar in the position indicated by the number
Windows key + B	Select the first item in the Notification Area and then use the arrow keys to cycle through the items Press Enter to open the selected item
Windows key + Ctrl + B	Access the program that is displaying a message in the Notification Area
Windows key + T	Cycle through the items on the Taskbar
Windows key + M	Minimize all windows
Windows key + Shift + M	Restore all minimized windows
Windows key + D	Show/Hide Desktop (minimize/restore all windows)
Windows key + L	Lock computer
Windows key + Up Arrow	Maximize current window
Windows key + Down Arrow	Minimize/restore current window
Windows key + Home	Minimize all but the current window
Windows key + Left Arrow	Tile window on the left side of the screen
Windows key + Right Arrow	Tile window on the right side of the screen
Windows key + Shift + Up Arrow	Extend current window from the top to the bottom of the screen
Windows key + Shift + Left/Right Arrow	Move the current window from one monitor to the next
Windows key + F1	Launch Windows Help and Support
Ctrl + Esc	Switch between Start screen and the last accessed application
Ctrl + Mouse scroll wheel	Activate the Semantic Zoom on the Modern Desktop screen
Alt	Display a hidden Menu Bar
Alt + D	Select the Address Bar
Alt + P	Display the Preview Pane in Windows Explorer
Alt + Tab	Cycle forward through open windows
Alt + Shift + Tab	Cycle backward through open windows
Alt + F	Opens file menu
Alt + Spacebar	Access the Shortcut menu for current window
Alt + Esc	Cycle between open programs in the order that they were opened
Alt + F4	Close Application
Alt + Enter	Open the Properties dialog box of the selected item
Alt + PrtScn	Take a screen shot of the active Window and place it in the clipboard
Alt + Up Arrow	Move up one folder level in Windows Explorer
Alt + Left Arrow	Display the previous folder
Alt + Right Arrow	Display the next folder
Shift + Insert	Load media disc without triggering Autoplay or Autorun
Shift + Delete	Permanently delete the item (rather than sending it to the Recycle Bin)
Shift + F6	Cycle backward through elements in a window or dialog box
Shift + F10	Access the context menu for the selected item

Shift + Tab	Cycle backward through elements in a window or dialog box
Shift + Click	Select a consecutive group of items
Shift + Click on a Taskbar button	Launch a new instance of a program
Shift + Right-click on a Taskbar button	Access the context menu for the selected item
Ctrl + A	Select all items
Ctrl + C	Copy the selected item
Ctrl + X	Cut the selected item
Ctrl + V	Paste the selected item
Ctrl + D	Delete selected item
Ctrl + Z	Undo an action
Ctrl + Y	Redo an action
Ctrl + N	Open a new window in Windows Explorer
Ctrl + W	Close current window in Windows Explorer
Ctrl + E	Select the Search box in the upper right corner of a window
Ctrl + Shift + N	Create new folder
Ctrl + Shift + Esc	Open the Task Manager
Ctrl + Alt + Tab	Use arrow keys to cycle through open windows
Ctrl + Alt + Delete	Access the Windows Security screen
Ctrl + Click	Select multiple individual items
Ctrl + Click and drag an item	Copies that item in the same folder
Ctrl + Shift + Click and drag an item	Creates a shortcut for that item in the same folder
Ctrl + Tab	Move forward through tabs
Ctrl + Shift + Tab	Move backward through tabs
Ctrl + Shift + Click on a Taskbar button	Launch a new instance of a program as an Administrator
Ctrl + Click on a grouped Taskbar button	Cycle through the instances of a program in the group
F1	Display Help
F2	Rename a file
F3	Open Search
F4	Display the Address Bar list
F5	Refresh display
F6	Cycle forward through elements in a window or dialog box
F7	Display command history in a Command Prompt
F10	Display hidden Menu Bar
F11	Toggle full screen display
Tab	Cycle forward through elements in a window or dialog box
Home	Move to the top of the active window or text line
End	Move to the bottom of the active window
Delete	Delete the selected item
Backspace	Display the previous folder in Windows Explorer Move up one folder level in Open or Save dialog box
Esc	Close a dialog box
Num Lock Enabled + Plus (+):	Display the contents of the selected folder
Num Lock Enabled + Minus (-):	Collapse the selected folder
Num Lock Enabled + Asterisk (*):	Expand all subfolders under the selected folder
Shift 5 times	Turn StickyKeys on or off
Hold down right Shift for 8 seconds	Turn FilterKeys on or off
Hold down Num Lock for 5 seconds	Turn ToggleKeys on or off
CTRL+MAIUSC+DEL	Access active account manager
CTRL+MAIUSC+ESC	Access task manager
(hold down) R-CTRL + *(twice)* SCROLL LOCK	Crash the system (read the section Force a System Crash using Keyboard)

Free activate a clean W10 install from an older version update

First you MUST run an upgrade of Windows 10 over your Genuine Windows 7 or 8.1, ensure 10 is online and activated after the upgrade, then you can clean install Windows 10 from USB or any other method, without the need of a product key. Once the clean install boot is completed and online, it should automatically activate Windows 10.

Windows 7 Pro/Ultimate => Windows 10 Pro

Windows 7 Home Basic \ Home Premium => Windows 10 Home

Windows 8.1 Pro => Windows 10 Pro

Windows 8.1 Core => Windows 10 Home

Check Windows Version

From CMD launch ***winver***, if you want to check the build String launch:

reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion" | findstr BuildLabEx

Check Performance Index

From CMD:

Get-WmiObject -Class Win32_WinSAT

Check License and Product Key

From Elevated CMD…
To check license type:
slmgr /dlv
to check license expiration date:
slmgr /xpr
To see the product key:
wmic path softwarelicensingservice get OA3xOriginalProductKey

Create your own Windows 10 ISO / Install Media

Download and run the Media Creation Tool from here:

https://www.microsoft.com/en-us/soft...load/windows10

This tool allows to create for free a copy of Windows 10 Pro or Home. If you haven't a product key you can install the copy using a "generic setup product keys".
You can find the list of product keys are on the official Microsoft Technet Website.
However, you must activate your copy. If you have a pc with a preinstalled UEFI license Windows should automatically detect it.

Backup of All Drivers

From Elevated Power Shell:

Export-WindowsDriver -Online -Destination D:\DriverBackup

If you want to export the driver from an offline image

Export-WindowsDriver -Path c:\offline-image -Destination D:\DriverBackup

Disable Defender (Pro and Enterprise)

From Elevated CMD:

gpedit.msc

Computer configuration > Administrative Templates > Windows Components > Windows Defender > Turn Off Windows Defender > Enable that setting

Hide Updates

Uninstall the offending update or driver, then run "Show or hide updates troubleshooter" ("*wushowhide.diagcab*").
https://support.microsoft.com/en-us/kb/3073930

Prevent Automatic Updates (Pro and Enterprise)

From Elevated CMD:

gpedit.msc

Computer configuration > Administrative Templates > Windows Components > Windows Update > Configure Automatic Updates > Enable that setting > Choose Option 2 (Notify for download and notify for install)

Delete Windows.old folder after upgrading

Right Click your C: > Properties > Disk Clean-up > Clean-up System Files

Bring back F8 Safemode options

From Elevated CMD:

Enable:

bcdedit /set {default} bootmenupolicy legacy

Disable:

bcdedit /set {default} bootmenupolicy standard

Advanced startup options

1) Open the Start menu and move your mouse pointer to the Shutdown button. Click it to open the Shutdown menu:

2) Press and hold the Shift key on the keyboard. Do not release the Shift key and click the Restart item:

3) Windows 10 will restart quickly and the Advanced Startup Options screen will appear

Alternative way:

1) Open the Start menu and click Settings.

2) Go to Update and recovery -> Recovery

3) There you will find "*Advanced startup*". Click the Restart now button.

In Advanced options menu, you can see also UEFI Firmware settings

Some application won't start

To run almost all compatible Windows application (if no require an exclusive proprietary software component) you need install all latest Visual C++ Redistributable you must have only an x86 and an x64 version for each VC++ release e.g.:

Microsoft Visual C++ 2005 x86
Microsoft Visual C++ 2005 x64
Microsoft Visual C++ 2008 x86
Microsoft Visual C++ 2008 x64
Microsoft Visual C++ 2010 x86
Microsoft Visual C++ 2010 x64
Microsoft Visual C++ 2012 x86
Microsoft Visual C++ 2012 x64
Microsoft Visual C++ 2013 x86
Microsoft Visual C++ 2013 x64
Microsoft Visual C++ 2015-2019 x86
Microsoft Visual C++ 2015-2019 x64

Install the *"NetFramework"* and *DirectX* runtimes and *MSXML*

Make sure you have all the latest drivers and windows updates

If you still have problems as alternative way you can try to Disable Data Execution Prevention but this procedure lowers the security level.

Adding app from commandline

Parameter Set: AddSet

Add-AppxPackage [-Path] <String> [-DependencyPath <String[]>] [-ForceApplicationShutdown] [-Confirm] [-WhatIf] [<CommonParameters>]

Parameter Set: RegisterSet

Add-AppxPackage [-Path] <String> -Register [-DependencyPath <String[]>] [-DisableDevelopmentMode] [-ForceApplicationShutdown] [-Confirm] [-WhatIf] [<CommonParameters>]

Parameter Set: UpdateSet

Add-AppxPackage [-Path] <String> -Update [-DependencyPath <String[]>] [-ForceApplicationShutdown] [-Confirm] [-WhatIf] [<CommonParameters>]

Removing apps (also the bundled) in Windows 10

To get the installed apps packages from Elevated CMD:

Get-AppXPackage

Removes an app package (.appx) from a user account.

Parameter Set: RemoveByPackageSet

Remove-AppxPackage [-Package] <String> [-Confirm] [-WhatIf] [<CommonParameters>]

e.g to remove myapp1_1.0.0.0_neutral__8wekyb3d8bbwe

Remove-AppxPackage myapp1_1.0.0.0_neutral__8wekyb3d8bbwe

Type the following command to remove all Modern apps from the system account:

Get-AppXProvisionedPackage -online | Remove-AppxProvisionedPackage -online

This means that all newly created user accounts will come without built-in Modern apps. This also means that new user accounts will be created faster.

Type the following command to remove all Modern apps from your current user accountstrong>:

Get-AppXPackage | Remove-AppxPackage

to remove all Metro apps from a specific user account.

Get-AppXPackage -User <username> | Remove-AppxPackage

remove Metro apps for all user accounts:

Get-AppxPackage -AllUsers | Remove-AppxPackage

Apps logging

Get the last error reported in the app package installation logs. From Elevated CMD:

Get-AppxLastError [<CommonParameters>]

Gets an app package (.appx) installation log.

Parameter Set: All

Get-AppxLog [-All] [<CommonParameters>]

Parameter Set: ActivityId

Get-AppxLog [-ActivityId <String>] [<CommonParameters>]

Gets the logs associated with the most recent deployment operation.

Get-AppxLog

Gets all the app package installation logs on the computer.

Get-AppxLog –All

Clear Metro App Cache

To clear app cache you can re-register all apps in this way, From Elevated CMD:

Get-AppXPackage | Foreach {Add-AppxPackage -DisableDevelopmentMode -Register "$($_.InstallLocation)\AppXManifest.xml"}

Fix damaged OS files

From Elevated CMD:

sfc /scannow

If "sfc" isn't able to repair all files use Restore DISM commands (Deployment Image Servicing and Management)

Deployment Image Servicing and Management (DISM) command

DISM is a command-line tool used to service Windows images offline before deployment. Subsets of the DISM servicing commands are also available for servicing a running operating system to recovery damaged system packages.

Restore DISM commands

From Elevated CMD:

dism /online /cleanup-image /checkhealth

dism.exe /online /cleanup-image /scanhealth

Dism.exe /Online /Cleanup-Image /AnalyzeComponentStore

Cleanup DISM commands

Dism /Online /Cleanup-Image /RestoreHealth

To remove update unistallation and recovery info

dism /online /cleanup-image /startcomponentcleanup

or the deepest version

dism /online /cleanup-image /startcomponentcleanup /ResetBase

by default "dism" uses Windows Update to get the system files, however you can use also a custom source, like Windows installation support, if for some reason isn't able to access the default source (in this sample a DVD in the drive E)

Dism /Online /Cleanup-Image /RestoreHealth /Source:wim:E:\sources\install.wim:1 /limitaccess

If you need to get details about what images are inside the ESD file you can use the command below. Note that ***/index:1*** is needed to get more detailed info about a specific version in case of distribution with multiple editions;

dism /Get-WimInfo /WimFile:E:\sources\install.esd /index:1

The next command allows you to convert the *"esd"* file in a *"wim"* file, "wim" files are used in older Windows distribution but could be more friendly if you need to use Windows Deployment Server (WDS);

dism /export-image /SourceImageFile:install.esd /SourceIndex:1 /DestinationImageFile:install.wim /Compress:max /CheckIntegrity

Adding a program to default open with menu

We want to add for the “7z” file type a new default program for open with menu, so we have to edit system register with regedit looking for ***.7z*** in the following path:
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts

expand the folder to open the ***OpenWithList***

Notice right away there is the same number of entries that are also found in the right-click menu. This is no coincidence, as this is where Windows looks to find these entries. We want to add a new additional entry so we can open 7z files with WinRAR. This executable is “WinRAR.exe,” so make a new “String Value” called the next letter in the list (eg ***d*** if we already have ***a*** end ***b***) with the executable name as the value, editing the string ***MRUList*** including the new letter created.

in

[HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\ApplicationAssociationToasts]

add dword32

Applications\WinRAR.exe_.7z

in

[HKEY_CLASSES_ROOT\Local Settings\Software\Microsoft\Windows\Shell\MuiCache]

each program must have 2 strings ***FriendlyAppName*** and ***ApplicationCompany*** that point to the path of executable in this form:

C:\Program Files\WinRAR\WinRAR.exe.FriendlyAppName

C:\Program Files\WinRAR\WinRAR.exe.ApplicationCompany

if one or both are missing create them. Now restarting the system, the menu open with should show the new application.

Adding shell to right click menu to open it in current position

If you want to add a shell to right click, so that It will point directly in the path where you have used the right click, add to system register these data:

```
[HKEY_CLASSES_ROOT\Directory\Background\shell\yourshell]
@="Your shell description"
"Icon"="C:\\YourIconPath\\yourshellicon.ico"

[HKEY_CLASSES_ROOT\Directory\Background\shell\yourshell\command]
@="C:\\YourShellPath\\yourshell.exe"
```

Fix Explorer crashes

The main cause of Explorer instability are the shell extensions installed from third part programs.

Try to disable non-Microsoft shell extension that most likely are causing problems using

ShellExViewer from Nirsoft http://www.nirsoft.net/ do right click on the extensions that you want

to disable and select "Disable Selected Item"

Fix Corrupted Icon Cache

If the icons are not displayed correctly probably the icon cache is corrupted. In this case you need to rebuild the Icon cache.
Open File Explorer > Folder Options > Views to show Hidden System Files. Next, go to ***C:\Users\YOUR_USERNAME\AppData\Local*** folder and delete the hidden file ***IconCache.db***. Reboot.
This action will purge and rebuild the icon cache.

Alternative way if the above method not work for you.

Open the command prompt as admin, type each of the following and after every command, hit the Enter button:

```
cd /d %userprofile%\AppData\Local
attrib –h IconCache.db
del IconCache.db
start explorer
```

Your Windows Icon Cache would have been rebuilt.

Fix Windows Media Player Library

Go to the folder

C:\Users\ YOURUSERACCOUNT \AppData\Local\Microsoft\Media Player

Delete the file

"***CurrentDatabase_xxx.wmdb***"

in

C:\Users\YOURUSERACCOUNT\AppData\Local\Microsoft\Media Player\Cache copertina\LocalMLS

delete album covers.

Fix Windows Recent File Broken

Remove the folder

C:\Users\ YOURUSERACCOUNT \AppData\Roaming\Microsoft\Windows\Recent

Restart the computer and from command line launch the command

shell:recent

Fix Missing Optical Units

From Elevated CMD:

```
reg.exe add "HKLM\System\CurrentControlSet\Services\atapi\Controller0" /f /v EnumDevice1 /t REG_DWORD /d 0x00000001
```

Alternative way

Delete the files "***cdr4_xp.sys***" and "***cdralw2k.sys***"

Open regedit

Go to the key

HKEY_LOCAL_MACHINE -> SYSTEM -> CurrentControlSet -> Control -> Class -> {4D36E965-E325-11CE-BFC1-08002BE10318}

Remove the strings "*Upper Filters*" and "*Lower Filters*"

Restart the PC

Fix Tray Icon Area

Open regedit to access system registry and go to

HKEY_CURRENT_USER\Software\Classes\Local Settings\Software\Microsoft\Windows\CurrentVersion\TrayNotify

Here delete

"***CurrentIconStream***" and "***PastIconsStream***"

Close the editor and restart Explorer.exe killing and start again the process from task manager

Fix Metro Apps

From command prompt as admin launch these commands

reset store

wsreset.exe

or

powershell -ExecutionPolicy Unrestricted Add-AppxPackage -DisableDevelopmentMode -Register ***$Env:SystemRoot\WinStore\AppxManifest.xml***

start "" "ms-windows-store:"

as alternative way you can clear metro app cache or use Microsoft Apps-troubleshooter

http://windows.microsoft.com/en-US/windows-8/what-troubleshoot-problems-app

Fix Windows update Problems

From Elevated CMD:

regsvr32.exe atl.dll

regsvr32.exe urlmon.dll

regsvr32.exe mshtml.dll

regsvr32.exe shdocvw.dll

regsvr32.exe browseui.dll

regsvr32.exe jscript.dll

regsvr32.exe vbscript.dll

regsvr32.exe scrrun.dll

regsvr32.exe msxml.dll

regsvr32.exe msxml3.dll

regsvr32.exe msxml6.dll

regsvr32.exe actxprxy.dll

regsvr32.exe softpub.dll

regsvr32.exe wintrust.dll

regsvr32.exe dssenh.dll

regsvr32.exe rsaenh.dll

regsvr32.exe gpkcsp.dll

regsvr32.exe sccbase.dll

regsvr32.exe slbcsp.dll

regsvr32.exe cryptdlg.dll

regsvr32.exe oleaut32.dll

regsvr32.exe ole32.dll

regsvr32.exe shell32.dll

regsvr32.exe initpki.dll

regsvr32.exe wuapi.dll

regsvr32.exe wuaueng.dll

regsvr32.exe wuaueng1.dll

regsvr32.exe wucltui.dll

regsvr32.exe wups.dll

regsvr32.exe wups2.dll

regsvr32.exe wuweb.dll

regsvr32.exe qmgr.dll

regsvr32.exe qmgrprxy.dll

regsvr32.exe wucltux.dll

regsvr32.exe muweb.dll

regsvr32.exe wuwebv.dll

In some cases, Windows cannot complete updates due to residual keys of deleted user accounts open regedit and go to

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList

Here remove the keys belonged to users that no longer existed in the system (DO NOT REMOVE KEYS RELATED TO ACTIVE USERS).

Restart the system and try to update again

Launch System Restore from command line

From Elevated CMD:

rstrui

Disable Microsoft Telemetry

Windows include many services that collect and send to Microsoft some data for Microsoft Improvement program, if you don't want these service safe resources or for privacy concern you have to:

1) In in ***Settings*** -> ***Privacy*** -> ***Feedback and Diagnostics*** set ***Diagnostic and Usage Data*** to ***Basic***

2) Open regedit and go to the following Registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\DataCollection

create a new a 32-bit DWORD value named ***AllowTelemetry*** and set it to ***0***.

2) Go to ***Services and Applications*** -> ***Services*** in the left pane. In the services list, disable the following services:

Diagnostics Tracking Service

dmwappushsvc

Connected User Experiences and Telemetry

Dmwappushsvc

2) Open Microsoft Task Scheduler and disable:

\Microsoft\Windows\Application Experience

Microsoft Compatibility Appraiser

\Microsoft\Windows\Autochk

Proxy

\Microsoft\Windows\RemoteAssistance

RemoteAssistanceTask

\Microsoft\Windows\Windows Error Reporting

Windows Error Reporting

\Microsoft\Windows\ErrorDetails

EnableErrorDetailsUpdate

ErrorDetailsUpdate

\Microsoft\Windows\Customer Experience Improvement Program

Consolidator

KernelCeipTask

UsbCeip

2) Optionally if you want to prevent the access to Microsoft telemetry servers add to your host file these lines:

127.0.0.1 vortex.data.microsoft.com

127.0.0.1 vortex-win.data.microsoft.com

127.0.0.1 telecommand.telemetry.microsoft.com

127.0.0.1 telecommand.telemetry.microsoft.com.nsatc.net

127.0.0.1 oca.telemetry.microsoft.com

127.0.0.1 oca.telemetry.microsoft.com.nsatc.net

127.0.0.1 sqm.telemetry.microsoft.com

127.0.0.1 sqm.telemetry.microsoft.com.nsatc.net

127.0.0.1 watson.telemetry.microsoft.com

127.0.0.1 watson.telemetry.microsoft.com.nsatc.net

127.0.0.1 redir.metaservices.microsoft.com

127.0.0.1 choice.microsoft.com

127.0.0.1 choice.microsoft.com.nsatc.net

127.0.0.1 df.telemetry.microsoft.com

127.0.0.1 reports.wes.df.telemetry.microsoft.com

127.0.0.1 services.wes.df.telemetry.microsoft.com

127.0.0.1 sqm.df.telemetry.microsoft.com

127.0.0.1 telemetry.microsoft.com

127.0.0.1 watson.ppe.telemetry.microsoft.com

127.0.0.1 telemetry.appex.bing.net

127.0.0.1 telemetry.urs.microsoft.com

127.0.0.1 telemetry.appex.bing.net:443

127.0.0.1 settings-sandbox.data.microsoft.com

127.0.0.1 vortex-sandbox.data.microsoft.com

127.0.0.1 telemetry.*

Check your IP Configuration

From CMD:

Ipconfig /all

This display full configuration information.

Renew IP Configuration

From Elevated CMD:

ipconfig /release
Release the IP address for the specified adapter.

ipconfig /renew
Renew the IP address for the specified adapter.

Reset IP Interface Configuration

From Elevated CMD:

netsh interface ipv4 reset
Reset the IPv4 interface.

Reset the IPv6 interface.

Reset Winsock Connection

In some cases, a wrong installation or removal of security software or malware etc. can broke the winsock configuration preventing the connection to internet of all programs that use this TCP/IP library. In this situation, you need to restore winsock configuration to the original status.
From Elevated CMD:

netsh winsock reset

after this command some third parts firewall may stop working and need reinstallation

Flush DNS Cache

The system dns resolver uses a cache to speedup domain name resolution. In some case e.g. domain setting have changed by few time the system may use old dns value due to this cache and this could lead to problems in accessing website. To force a dns cache refresh, you can flush the current values with this command:

ipconfig /flushdns

Connection Limits

By default, Windows use a connection limit. To Remove the limit open "regedit" and navigate to:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters

Find and open "***EnableConnectionRateLimiting***" to set the value data to zero. If this attribute is missing use right click and create a DWORD (32 bit) named "***EnableConnectionRateLimiting***" with value ***0*** check the
ipconfig /flushdns

You can also use group policy editor for

Computer Configuration\Administrative Templates\Network\Windows Connection Manager

and disable ***Minimize the number of simultaneous connections***

for info about server limit open "cmd" as admin and execute the command: ***net config server***

Cached Logon Limits

By default, Windows use a cached logon limit of 10. This determines how many user account entries Windows saves in the logon cache on the local computer. Windows saves the user account data that is used to log on to the computer so the data can be used if the user's domain controller is unavailable.

If the value of this entry is 0, Windows does not save any user account data in the logon cache. In that case, if the user's domain controller is not available and a user tries to log on to a computer that does not have the user's account information, Windows displays the following message: "*The system cannot log you on now because the domain <Domain-name> is not available*. You can change this value editing with regedit:

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\CachedLogonsCount

Register and Unregister a DLL

You can register or unregister DLL components using "regsvr32". This command-line tool registers .dll files as command components in the registry.

Syntax:
regsvr32 [/u] [/s] [/n] [/i[:cmdline]] dllname

Parameters:

/u : Unregisters server.

/s : Specifies regsvr32 to run silently and to not display any message boxes.

/n : Specifies not to call DllRegisterServer. You must use this option with /i.

/i :cmdline : Calls DllInstall passing it an optional [cmdline]. When used with /u, it calls dll uninstall.

dllname : Specifies the name of the dll file that will be registered.

/? : Displays help at the command prompt.

In 64bit version of Windows the path where are located the most system DLL libraries is **C:\Windows\System32** for 64 bit DLL and ***C:\Windows\SysWOW64*** for the 32 bit DLL. If you invoke ***regsvr32*** for a library located in one of these paths the OS switch automatically to the 64 or 32 bit version of ***regsvr32***, for this reason when you copy a DLL in one of these folders you have to pay attention that are placed in the correct folder. In the 32 bit version of Windows there is only **C:\Windows\System32** and is used for 32 bit DLL.

Hyper-V -V ON-OFF

From Elevated CMD:

Hyper-V OFF:

bcdedit /set hypervisorlaunchtype off

Hyper-V ON:

bcdedit /set hypervisorlaunchtype auto

To remove Hyper-V

dism.exe /Online /Disable-Feature:Microsoft-Hyper-V

To install Hyper-V

dism.exe /Online /Enable-Feature:Microsoft-Hyper-V /All

Hibernate ON-OFF

Hibernate mode ON:

powercfg -hibernate on

Hibernate mode OFF:

powercfg -hibernate off

Note: if you have enabled hibernate but you can see the command, you may need to disabled "***allow hybrid sleep***" in power management options in control panel.

Prevent USB Device sleep

In the Edit Plan Settings dialog click "Change advanced power settings" and disable USB sleep. Go to "Control Panel" -> "Device Manager", locate the device and in the power tab, uncheck "allow the computer to turn off this device" (if this section is present). In case of External USB hard drive, you may need addition steps to prevent sleep due to not perfect driver implementation in some USB host interfaces.

1) Go to "Control Panel" -> "Device Manager" expand the Disk Drives section in the USB Mass Storage Device node, locate the device, right click on it and select Properties.

2) On the Details tab, select "Parent" to identify host hardware id from the drop-down list. If you are in the right section you should see something like ***USB\VID_0004&PID_0001&REV_0001***
Make a note of the 4 digits occurring after "***VID_***" and "***PID_***" and close Device Manager. In this example, VID is ***0004*** and PID is ***0001***.

3) Run Registry Editor (regedit.exe) as administrator by searching for ***regedit*** in the Search charm. Right-click the regedit icon, and select Run as administrator.

4) In Registry Editor, navigate to ***HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\usbstor***.
Right-click "usbstor" and create a key. The name must be ***VID*** and ***PID*** strings obtained previously. For the device in this example the new key name is ***00040001***.

5) Right-click the new key and create a *DWORD32* entry named ***DeviceHackFlags***. Set the value to ***400***. Restart the system.

Temp System Folders Cleanable

You can empty

C:\Users\Silvio\AppData\Local\Temp

You can empty also:

C:\Windows\SoftwareDistribution\Download

You can delete *$SysReset*, *$Windows~BT* and *$Windows~WS* temporary directories used by windows to update your system

From Elevated CMD:

```
takeown /F C:\$Windows.~BT\* /R /A
icacls C:\$Windows.~BT\*.* /T /grant administrators:F
rmdir /S /Q C:\$Windows.~BT\

takeown /F C:\$Windows.~WS\* /R /A
icacls C:\$Windows.~WS\*.* /T /grant administrators:F
rmdir /S /Q C:\$Windows.~WS\
```

You can also reduce the size of WinSxS (the archive component folder of Windows) using the DISM clean commands.

NOTE: if you clean update folders you cannot undo updates. So before do operations in the next section make sure all updates are installed correctly.

Enable Trim

Trim allows operating system to inform a solid-state drive (SSD) which blocks of data are no longer considered in use and can be wiped internally. To check status launch from CMD:

fsutil behavior query DisableDeleteNotify
If ***DisableDeleteNotify = 0*** , trim is enabled, if It's 1, then if you have an SSD you should enable It with:

fsutil behavior set DisableDeleteNotify 0

Scandisk from the Command-Line for More Complete Control

chkdsk C: /f /r /b /x

/f parameter tells CHKDSK to fix any errors it finds;

/r tells it to locate the bad sectors on the drive and

recover readable information; "/x" forces the drive to dismount

before the process starts.

/b Clears the list of bad clusters on the volume and rescans

all allocated and free clusters for errors. /b includes the

functionality of /r. Use this parameter after imaging a volume

to a new hard disk drive.

/v Displays the name of each file in every directory as the disk is checked.

/i Use with NTFS only. Performs a less vigorous check of index entries,

which reduces the amount of time required to run chkdsk.

/c Use with NTFS only. Does not check cycles within the folder structure,

which reduces the amount of time required to run chkdsk

Exit codes:

0 = No errors were found.

1 = Errors were found and fixed.

2 = Performed disk cleanup (such as garbage collection) or did not perform cleanup because /f was not specified.

3 = Could not check the disk, errors could not be fixed, or errors were not fixed because /f was not specified.

Defragmentation from the Command-Line for More Complete Control

Defrag <volume> | /C | /E <volumes> [/A | /X | /T] [/H] [/M] [/U] [/V]

<volume> is the drive letter or mount point of the volume to defragment, the options for ***Defrag.exe*** are:

/C Defragment all local volumes on the computer.

/E Defragment all local volumes on the computer except those specified.

/A Display a fragmentation analysis report for the specified volume without defrag¬menting it.

/X Perform free-space consolidation. Free-space consolidation is useful if you need to shrink a volume, and it can reduce fragmentation of future files.

/T Track an operation already in progress on the specified volume.

/H Run the operation at normal priority instead of the default low priority. Specify this option if a computer is not otherwise in use.

/M Defragment multiple volumes simultaneously, in parallel. This is primarily useful for computers that can access multiple disks simultaneously, such as those using SCSI- or SATA-based disks rather than disks with an IDE interface.

/U Print the progress of the operation on the screen.

/V Verbose mode. Provides additional detail and statistics.

NOTE
To perform a perfect defrag you should disable hibernation before the procedure and enable it again at the end (Hibernation create a file as large as your RAM memory on hard disk that cannot be easily defragmented).

Secure Boot Checks

From power shell started as admin

Confirm-SecureBootUEFI

Confirms that Secure Boot is enabled by checking the Secure Boot status on the local computer.

Format-SecureBootUEFI

Formats certificates or hashes into a content object that is returned and creates a file that is ready to be signed.

Get-SecureBootPolicy

Gets the publisher GUID and the policy version of the Secure Boot configuration policy.

Get-SecureBootUEFI

Gets the UEFI variable values related to Secure Boot such as the SetupMode, SecureBoot, KEK, PK, SignatureDatabase, and forbidden SignatureDatabase.

Set-SecureBootUEFI

Sets the Secure Boot-related UEFI variables such as Platform Key, Key Exchange Key, Signature Database and Forbidden Signature Database.

Memory Integrity Checks

To identify memory problems on your computer you can run Windows Memory Diagnostics Tool
From Command Prompt type:
mdsched
Will be shown a message that ask you to restart Windows to perform a memory test or schedule it at next restart.

Force a System Crash using Keyboard

You must ensure the following settings before the keyboard can cause a system crash:

-With PS/2 keyboards, you must enable the keyboard-initiated crash in the registry. In the registry key ***HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\i8042prt\Parameters, create a value named CrashOnCtrlScroll, and set it equal to a REG_DWORD value of 0x01***.

-With USB keyboards, you must enable the keyboard-initiated crash in the registry. In the registry key ***HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\kbdhid\Parameters, create a value named CrashOnCtrlScroll, and set it equal to a REG_DWORD value of 0x01***.

If you wish a crash dump file to be written, you must enable such dump files, choose the path and file name, and select the size of the dump file. For more information, see Dump File Creation.

You must restart the system for these settings to take effect.

After this is completed, the keyboard crash can be initiated by using the following hotkey sequence:
Hold down the rightmost **CTRL** key, and press the **SCROLL LOCK** key twice.

Wipe drive using MBR or GPT partition style

If you want a clean install of Windows

1) Turn off the PC, and put in the Windows installation DVD or USB key.

2) Boot the PC to the DVD or USB key in UEFI mode. For more info, see Boot to UEFI Mode or Legacy BIOS mode.

3) From inside Windows Setup, press Shift+F10 to open a command prompt window.

4) Open the "diskpart" tool:
diskpart

5) Identify the drive to reformat:
list disk

6) Select the drive, and reformat it:
select disk <disk number>
clean
convert gpt (only if you want to use UEFI mode)
exit

7) Close the command prompt window.

8) Continue the Windows Setup installation. When choosing an installation type, select Custom. The drive will appear as a single area of unallocated space. Select the unallocated space and click Next. Windows begins the installation.

Dump File Creation

To enable the creation of kernel mode dump file in case of crash go to Control Panel > System and Security > System. Click Advanced system settings. Under Startup and Recovery, click Settings. Under Write Debugging Information, you can specify a kernel-mode dump file setting. Only one dump file can be created for any given crash.

The Complete Memory Dump is the largest and contains the most information, the Kernel Memory Dump is somewhat smaller, and the Small Memory Dump is only 64 KB in size.

If you select Automatic Memory Dump, the dump file is the same as a Kernel Memory Dump, but Windows has more flexibility in setting the size of the system paging file.

The advantage to the larger files is that, since they contain more information, they are more likely to help you find the cause of the crash.

The advantage of the smaller files is that they are smaller and written more quickly. Speed is often valuable; if you are running a server, you may want the server to reboot as quickly as possible after a crash, and the reboot will not take place until the dump file has been written.

You can generate dump files also with the ***.dump*** command you can use the parameter ***-?*** to see details about the command and related options.

Batch Script Creation

Batch scripts are text files that carry out a series of commands. These files have the special extension BAT or CMD. Files of this type are recognized and executed through the command interpreter of system shell.
To create a batch file is enough create a text file with commands supported by command prompt and rename this file extension in ***.bat***.

Basic batch commands table

Command	Description
.	Represents the default directory.
..	Represents the parent of the default directory.
@	Hide the current command line output.
%NUMBER	The number of parameter. %1 represents the first string typed after the batch commmand. Each occurrence of %digit is replaced by the corresponding string from the batch command line. Examples: ***MYBATCH jpg D:*** ***COPY *.%1 %2*** Copies all .jpg files in the default directory to drive D.
%VARIABLE%	Replaces the environment variable name with its environment value. Examples: ***echo %PATH%*** Displays the value of PATH, the current search path.
> >> < <<	Redirects directives. Syntax: ***command > nul*** Redirects command output to oblivion. ***command > file*** Redirects command output to file. ***command >> file*** Appends command output to file. ***command < file*** Redirects file output to command.
CALL	Loads and executes a batch file within a batch file as external command. When a second batch file completes, control returns to the calling file.
CD	Displays the current directory and its path.
CLS	Clears the video display screen.
COPY	Copy file and directories. Syntax: ***copy source destination***
DATE	Displays the current system date setting. Used without parameters, date displays the current system date setting and prompts you to type a new date. Syntax: ***date [mm-dd-yy] [/t]***
DEL	Remove files and directories. Important parameters for this command are ***/p*** : Prompts you for confirmation before deleting the specified file. ***/f*** : Forces deletion of read-only files.

	/s : Deletes specified files from the current directory and all subdirectories. Displays the names of the files as they are being deleted. ***/q*** : Specifies quiet mode. You are not prompted for delete confirmation. ***/a*** : Deletes files based on specified attributes. Syntax: ***del target***
ECHO	Controls whether commands and comments within a batch file are displayed. Examples: ***@echo OFF*** *Hides the commands output.* ***@echo ON*** Restores normal display activity.
EXIT	Exits the current batch script.
FC	Compares two files and displays the differences between them. Syntax: ***fc [/a] [/b] [/c] [/l] [/lbn] [/n] [/t] [/u] [/w] [/nnnn] [drive1:][path1]filename1 [drive2:][path2]filename2*** Parameters: ***/a*** : Abbreviates the output of an ASCII comparison. Instead of displaying all of the lines that are different, fc displays only the first and last line for each set of differences. ***/b*** : Compares the files in binary mode. Fc compares the two files byte by byte and does not attempt to resynchronize the files after finding a mismatch. This is the default mode for comparing files that have the following file extensions: .exe, .com, .sys, .obj, .lib, or .bin. ***/c*** : Ignores the case of letters. ***/l*** : Compares the files in ASCII mode. Fc compares the two files line by line and attempts to resynchronize the files after finding a mismatch. This is the default mode for comparing files, except files with the following file extensions: .exe, .com, .sys, .obj, .lib, or .bin. ***/lb n*** : Sets the n number of lines for the internal line buffer. The default length of the line buffer is 100 lines. If the files that you are comparing have more than this number of consecutive differing lines, fc cancels the comparison. ***/n*** : Displays the line numbers during an ASCII comparison. ***/t*** : Prevents fc from converting tabs to spaces. The default behavior is to treat tabs as spaces, with stops at each eighth character position. ***/u*** : Compares files as Unicode text files. ***/w*** : Compresses white space (that is, tabs and spaces) during the comparison. If a line contains many consecutive spaces or tabs, ***/w*** treats these characters as a single space. When used with the /w command-line option, fc ignores (and does not compare) white space at the beginning and end of a line. ***/ nnnn*** : Specifies the number of consecutive lines that must match before fc considers the files to be resynchronized. If the number of matching lines in the files is less than nnnn, fc displays the matching lines as differences. The default value is 2. Examples: ***fc /a monthly.rpt sales.rpt*** Shows the lines of monthly.rpt file that matches with sales.rpt file
FIND	Searches for a specific string. Syntax: ***find [/v] [/c] [/n] [/i] "string" [[Drive:][Path]FileName[...]]*** Parameters: ***/v*** : Displays all lines that do not contain the specified string. ***/c*** : Counts the lines that contain the specified string and displays the total. ***/n*** : Precedes each line with the file's line number. ***/i*** : Specifies that the search is not case-sensitive. Examples:

	find "Anna" employers.txt Find a string that contains text Anna in the file employers.txt
FOR	Loop operations. Syntax: ***for %%argument in (list) do command*** Examples: ***for %%d in (A,C,D) do DIR %%d *.**** Displays the directories of drives A, C, and D sequentially.
GOTO	Transfers control within a batch file to a line identified by a label. The label must be of the form ":LABEL". Syntax: ***goto LABEL*** ***:LABEL***
HELP	Provides online information about system commands. Used without parameters, help lists and briefly describes every system command. Syntax: ***help command***
HOSTNAME	Displays the host name portion of the full computer name of the computer.
IF	Tests a condition and executes a command only if the condition is true. Before the condition you can use ***[not]*** modifier to negate condition. Support also others modifiers like ***exist*** Syntax: ***if condition command*** Examples: ***if "%1"= ="" goto ERROR*** If there is no parameter, then control is transferred to label ERROR.
MKDIR	Creates a directory or subdirectory. Syntax: ***mkdir directory***
MORE	Displays one screen of output at a time. Syntax: ***command | more [/c] [/p] [/s] [/tn] [+n]*** ***more [[/c] [/p] [/s] [/tn] [+n]] < [Drive:] [Path] FileName*** ***more [/c] [/p] [/s] [/tn] [+n] [files]*** Parameters: ***/c*** : Clears screen before displaying page. ***/p*** : Expands form-feed characters. ***/s*** : Changes multiple blank lines to one blank line. ***/t n*** : Changes tabs to the number of spaces specified by n. ***+ n*** : Displays first file beginning at the line specified by n. ***files*** : Specifies list of files to display. Separate file names with a space.
MOVE	Move files and directories. Syntax: ***move source destination***
PAUSE	Pauses the running of a batch file and displays the message "Press any key to continue ..." on the screen. If the optional message is included, it will be displayed first. Syntax: ***pause [message]***
PING	Verifies IP-level connectivity to another TCP/IP computer by sending Internet Control Message Protocol (ICMP) Echo Request messages. The receipt of corresponding Echo Reply messages are displayed, along with round-trip times. Syntax: ***ping [-t] [-a] [-n Count] [-l Size] [-f] [-i TTL] [-v TOS] [-r Count] [-s Count] [{-j HostList | -k HostList}] [-w Timeout] [TargetName]***

	Parameters: ***-t*** : Specifies that ping continue sending Echo Request messages to the destination until interrupted. To interrupt and display statistics, press CTRL-BREAK. To interrupt and quit ping, press CTRL-C. ***-a*** : Specifies that reverse name resolution is performed on the destination IP address. If this is successful, ping displays the corresponding host name. ***-n Count*** : Specifies the number of Echo Request messages sent. The default is 4. ***-l Size*** : Specifies the length, in bytes, of the Data field in the Echo Request messages sent. The default is 32. The maximum size is 65,527. ***-f*** : Specifies that Echo Request messages are sent with the Don't Fragment flag in the IP header set to 1. The Echo Request message cannot be fragmented by routers in the path to the destination. This parameter is useful for troubleshooting path Maximum Transmission Unit (PMTU) problems. ***-i TTL*** : Specifies the value of the TTL field in the IP header for Echo Request messages sent. The default is the default TTL value for the host. For Windows XP hosts, this is typically 128. The maximum TTL is 255. ***-v TOS*** : Specifies the value of the Type of Service (TOS) field in the IP header for Echo Request messages sent. The default is 0. TOS is specified as a decimal value from 0 to 255. ***-r Count*** : Specifies that the Record Route option in the IP header is used to record the path taken by the Echo Request message and corresponding Echo Reply message. Each hop in the path uses an entry in the Record Route option. If possible, specify a Count that is equal to or greater than the number of hops between the source and destination. The Count must be a minimum of 1 and a maximum of 9. ***-s Count*** : Specifies that the Internet Timestamp option in the IP header is used to record the time of arrival for the Echo Request message and corresponding Echo Reply message for each hop. The Count must be a minimum of 1 and a maximum of 4. ***-j HostList*** : Specifies that the Echo Request messages use the Loose Source Route option in the IP header with the set of intermediate destinations specified in HostList. With loose source routing, successive intermediate destinations can be separated by one or multiple routers. The maximum number of addresses or names in the host list is 9. The host list is a series of IP addresses (in dotted decimal notation) separated by spaces. ***-k HostList*** : Specifies that the Echo Request messages use the Strict Source Route option in the IP header with the set of intermediate destinations specified in HostList. With strict source routing, the next intermediate destination must be directly reachable (it must be a neighbor on an interface of the router). The maximum number of addresses or names in the host list is 9. The host list is a series of IP addresses (in dotted decimal notation) separated by spaces. ***-w Timeout*** : Specifies the amount of time, in milliseconds, to wait for the Echo Reply message that corresponds to a given Echo Request message to be received. If the Echo Reply message is not received within the time-out, the "Request timed out" error message is displayed. The default time-out is 4000 (4 seconds). ***TargetName*** : Specifies the destination, which is identified either by IP address or host name.
PRINT	Sends a text file to a printer. Syntax: ***print [/d:Printer] [Drive:][Path] FileName [...]*** Parameters: ***/d: Printer*** : Specifies the printer on which you want to print the job. You can specify a local printer by specifying the port on your computer to which the printer is connected. Valid values for parallel ports are LPT1, LPT2, and LPT3. Valid values for serial ports are COM1, COM2, COM3, and COM4. You can also specify a network printer by its queue name (\\ServerName\ShareName). If you do not specify a printer, the print job is sent to LPT1. ***Drive :*** : Specifies the logical or physical drive on which the file you want to print is located. This parameter is not required if the file you want to print is located on the current drive.
REM	Adds remarks to a batch file. Syntax: ***rem [remark]***
RENAME	Rename a file Syntax: ***rename source target***
REPLACE	Replaces files in the destination directory with files in the source directory that have the same name. You can also use replace to add unique file names to the destination directory.

	Syntax: ***replace source target***
SET	Set is used to view, create, change, or delete environment values. Syntax: ***set [variable=[value]]*** Examples: ***set*** Display the entire DOS environment. **set USER=Silvio** Sets the value of USER to the string, "Silvio". **set USER=** Removes USER from the environment. ***set PATH=C:\;C:\MyFolder*** Sets C:\;C:\MyFolder as the current search path. ***set PATH=%PATH%;C:\TEST*** Appends ;C:\TEST to the current search path.
SHIFT	Shifts any parameter on the command line one position to the left. Use SHIFT to refer to multiple parameters by one name or to use more than ten parameters on a single command line. Examples: ***:LOOP*** ***COPY %1 D:*** ***shift*** ***if not (%1)==() goto LOOP*** Beginning with the first parameter, all the parameters listed on the command line are iterated and a file, the value of the parameter, is copied to D.
SORT	Sorts the input. Important attributes for this command are: ***/+n*** : Specifies the character number, n, to begin each comparison. /+3 indicates that each comparison should begin at the 3rd character in each line. Lines with fewer than n characters collate before other lines. By default, comparisons start at the first character in each line. ***/L*** : Overrides the system default locale with the specified one. ***/M*** : Specifies amount of main memory to use for the sort, in kilobytes. /REC : Specifies the maximum number of characters in a record ***/R*** : Reverses the sort order ***/T*** : Specifies the path of the directory to hold the sort's working storage ***/O*** : Specifies the file where the sorted input is to be stored. If not specified, the data is written to the standard output. Syntax: ***SORT [/R] [/+n] [/M kilobytes] [/L locale] [/REC recordbytes] [[drive1:][path1]filename1] [/T [drive2:][path2]] [/O [drive3:][path3]filename3]*** Examples: ***sort testfile.txt /o resultfile.txt*** Sort testfile.txt and place the result in resultfile.txt
START	Starts a separate Command Prompt window to run a specified program or command. Syntax: ***start ["title"] [/dPath] [/i] [/min] [/max] [{/separate | /shared}] [{/low | /normal | /high | /realtime | /abovenormal | belownormal}] [/wait] [/b] [FileName] [parameters]*** Parameters: ***" title "*** : Specifies the title to display in Command Prompt window title bar. ***/d Path*** : Specifies the startup directory. ***/I*** : Passes the Cmd.exe startup environment to the new Command Prompt window. ***/min*** : Starts a new minimized Command Prompt window.

	/max : Starts a new maximized Command Prompt window. ***/separate*** : Starts 16-bit programs in a separate memory space. ***/shared*** : Starts 16-bit programs in a shared memory space. ***/low*** : Starts an application in the idle priority class. ***/normal*** : Starts an application in the normal priority class. ***/high*** : Starts an application in the high priority class. ***/realtime*** : Starts an application in the realtime priority class. ***/abovenormal*** : Starts an application in the abovenormal priority class. ***/belownormal*** : Starts an application in the belownormal priority class. ***/wait*** : Starts an application and waits for it to end. ***/b*** : Starts an application without opening a new Command Prompt window. CTRL+C handling is ignored unless the application enables CTRL+C processing. Use CTRL+BREAK to interrupt the application.
TASKKILL	Terminate a task. Syntax: ***taskkill [/s Computer] [/u Domain\User [/p Password]]] [/fi FilterName] [/pid ProcessID]\[/im ImageName] [/f][/t]*** Examples: ***taskkill /pid 1230 /pid 1241 /pid 1253*** Kill the task with process id 1253. ***taskkill /f /fi "USERNAME eq NT AUTHORITY\SYSTEM" /im notepad.exe*** Kill the notepad task of specified user. ***taskkill /s srvmain /f /im notepad.exe*** Kill the notepad task of specified computer.
TAKEOWN	***takeown [/s <Computer> [/u [<Domain>\]<User name> [/p [<Password>]]]] /f <File name> [/a] [/r [/d {Y|N}]]*** ***/a*** Gives ownership to the Administrators group instead of the current user. ***/r*** Performs a recursive operation on all files in the specified directory and subdirectories. ***/d {Y | N}*** Suppresses the confirmation prompt that is displayed when the current user does not have the "List Folder" permission on a specified directory, and instead uses the specified default value. Valid values for the /d option are as follows: ***Y*** Take ownership of the directory. ***N*** Skip the directory. Note that you must use this option in conjunction with the ***/r*** option. Example: ***takeown /F " C:\Mypath\myfile" /A***
TASKLIST	Displays the list of applications and services with their Process ID (PID) for all tasks running on either a local or a remote computer. Syntax: ***tasklist[.exe] [/s computer] [/u domain\user [/p password]] [/fo {TABLE|LIST|CSV}] [/nh] [/fi FilterName [/fi FilterName2 [...]]] [/m [ModuleName] | /svc | /v]***
TREE	Graphically displays the directory structure of a path or of the disk in a drive. Syntax: ***tree [Drive:][Path] [/f] [/a]***

Default environment variables

Variables that are processed for the operating system and in the context of each user

You can use these variables within sections in the .xml files with ***context=UserAndSystem***, ***context=User***, and ***context=System***.

Variable	Explanation
ALLUSERSAPPDATA	Same as **CSIDL_COMMON_APPDATA**.
ALLUSERSPROFILE	Refers to %**PROFILESFOLDER**%\Public or %**PROFILESFOLDER**%\all users.
COMMONPROGRAMFILES	Same as **CSIDL_PROGRAM_FILES_COMMON**.
COMMONPROGRAMFILES**(X86)**	Refers to the C:\Program Files (x86)\Common Files folder on 64-bit systems.
CSIDL_COMMON_ADMINTOOLS	Version 10.0. The file-system directory that contains administrative tools for all users of the computer.
CSIDL_COMMON_ALTSTARTUP	The file-system directory that corresponds to the non-localized Startup program group for all users.
CSIDL_COMMON_APPDATA	The file-system directory that contains application data for all users. A typical path Windows is C:\ProgramData.
CSIDL_COMMON_DESKTOPDIRECTORY	The file-system directory that contains files and folders that appear on the desktop for all users. A typical Windows® XP path is C:\Documents and Settings\All Users\Desktop. A typical path is C:\Users\Public\Desktop.
CSIDL_COMMON_DOCUMENTS	The file-system directory that contains documents that are common to all users. A typical path in Windows XP is C:\Documents and Settings\All Users\Documents. A typical path is C:\Users\Public\Documents.
CSIDL_COMMON_FAVORITES	The file-system directory that serves as a common repository for favorites common to all users. A typical path is C:\Users\Public\Favorites.
CSIDL_COMMON_MUSIC	The file-system directory that serves as a repository for music files common to all users. A typical path is C:\Users\Public\Music.
CSIDL_COMMON_PICTURES	The file-system directory that serves as a repository for image files common to all users. A typical path is C:\Users\Public\Pictures.
CSIDL_COMMON_PROGRAMS	The file-system directory that contains the directories for the common program groups that appear on the **Start** menu for all users. A typical path is C:\ProgramData\Microsoft\Windows\Start Menu\Programs.
CSIDL_COMMON_STARTMENU	The file-system directory that contains the programs and folders which appear on the **Start** menu for all users. A typical path in Windows is C:\ProgramData\Microsoft\Windows\Start Menu.
CSIDL_COMMON_STARTUP	The file-system directory that contains the programs that appear in the Startup folder for all users. A typical path in Windows XP is C:\Documents and Settings\All Users\Start Menu\Programs\Startup. A typical path is C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup.
CSIDL_COMMON_TEMPLATES	The file-system directory that contains the templates that are available to all users. A typical path is C:\ProgramData\Microsoft\Windows\Templates.
CSIDL_COMMON_VIDEO	The file-system directory that serves as a repository for video files common to all users. A typical path is C:\Users\Public\Videos.
CSIDL_DEFAULT_APPDATA	Refers to the Appdata folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_LOCAL_APPDATA	Refers to the local Appdata folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_COOKIES	Refers to the Cookies folder inside %**DEFAULTUSERPROFILE**%.

CSIDL_DEFAULT_CONTACTS	Refers to the Contacts folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_DESKTOP	Refers to the Desktop folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_DOWNLOADS	Refers to the Downloads folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_FAVORITES	Refers to the Favorites folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_HISTORY	Refers to the History folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_INTERNET_CACHE	Refers to the Internet Cache folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_PERSONAL	Refers to the Personal folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_MYDOCUMENTS	Refers to the My Documents folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_MYPICTURES	Refers to the My Pictures folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_MYMUSIC	Refers to the My Music folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_MYVIDEO	Refers to the My Videos folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_RECENT	Refers to the Recent folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_SENDTO	Refers to the Send To folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_STARTMENU	Refers to the Start Menu folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_PROGRAMS	Refers to the Programs folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_STARTUP	Refers to the Startup folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_TEMPLATES	Refers to the Templates folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_DEFAULT_QUICKLAUNCH	Refers to the Quick Launch folder inside %**DEFAULTUSERPROFILE**%.
CSIDL_FONTS	A virtual folder containing fonts. A typical path is C:\Windows\Fonts.
CSIDL_PROGRAM_FILESX86	The Program Files folder on 64-bit systems. A typical path is C:\Program Files(86).
CSIDL_PROGRAM_FILES_COMMONX86	A folder for components that are shared across applications on 64-bit systems. A typical path is C:\Program Files(86)\Common.
CSIDL_PROGRAM_FILES	The Program Files folder. A typical path is C:\Program Files.
CSIDL_PROGRAM_FILES_COMMON	A folder for components that are shared across applications. A typical path is C:\Program Files\Common.
CSIDL_RESOURCES	The file-system directory that contains resource data. A typical path is C:\Windows\Resources.
CSIDL_SYSTEM	The Windows System folder. A typical path is C:\Windows\System32.
CSIDL_WINDOWS	The Windows directory or system root. This corresponds to the %**WINDIR**% or %**SYSTEMROOT**% environment variables. A typical path is C:\Windows.
DEFAULTUSERPROFILE	Refers to the value in **HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList [DefaultUserProfile]**.
PROFILESFOLDER	Refers to the value in **HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList [ProfilesDirectory]**.
PROGRAMFILES	Same as **CSIDL_PROGRAM_FILES**.
PROGRAMFILES(X86)	Refers to the C:\Program Files (x86) folder on 64-bit systems.
SYSTEM	Refers to %**WINDIR**%\system32.
SYSTEM16	Refers to %**WINDIR**%\system.

SYSTEM32	Refers to %WINDIR%\system32.
SYSTEMPROFILE	Refers to the value in HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList\S-1-5-18 [ProfileImagePath].
SYSTEMROOT	Refers to the root of the system drive.
WINDIR	Refers to the Windows folder located on the system drive.

Variables that are recognized only in the user context

You can use these variables in the .xml files within sections with context=User and context=UserAndSystem.

Variable	Explanation
APPDATA	Same as CSIDL_APPDATA.
CSIDL_ADMINTOOLS	The file-system directory that is used to store administrative tools for an individual user. The Microsoft® Management Console (MMC) saves customized consoles to this directory, which roams with the user profile.
CSIDL_ALTSTARTUP	The file-system directory that corresponds to the user's non-localized Startup program group.
CSIDL_APPDATA	The file-system directory that serves as a common repository for application-specific data. A typical path is C:\Documents and Settings\username\Application Data or C:\Users\username\AppData\Roaming.
CSIDL_BITBUCKET	The virtual folder that contains the objects in the user's Recycle Bin.
CSIDL_CDBURN_AREA	The file-system directory acting as a staging area for files waiting to be written to CD. A typical path is C:\Users\username\AppData\Local\Microsoft\Windows\MasteredBurning\Disc Burning.
CSIDL_CONNECTIONS	The virtual folder representing Network Connections that contains network and dial-up connections.
CSIDL_CONTACTS	This refers to the Contacts folder in %CSIDL_PROFILE%.
CSIDL_CONTROLS	The virtual folder that contains icons for the Control Panel items.
CSIDL_COOKIES	The file-system directory that serves as a common repository for Internet cookies. A typical path is C:\Users\username\AppData\Roaming\Microsoft\Windows\Cookies.
CSIDL_DESKTOP	The virtual folder representing the Windows desktop.
CSIDL_DESKTOPDIRECTORY	The file-system directory used to physically store file objects on the desktop, which should not be confused with the desktop folder itself. A typical path is C:\Users\username\Desktop.
CSIDL_DRIVES	The virtual folder representing My Computer that contains everything on the local computer: storage devices, printers, and Control Panel. The folder may also contain mapped network drives.
CSIDL_FAVORITES	The file-system directory that serves as a common repository for the user's favorites. A typical path is C:\Users\Username\Favorites.
CSIDL_HISTORY	The file-system directory that serves as a common repository for Internet history items.
CSIDL_INTERNET	A virtual folder for Internet Explorer.
CSIDL_INTERNET_CACHE	The file-system directory that serves as a common repository for temporary Internet files. A typical path is C:\Users\username\AppData\Local\Microsoft\Windows\Temporary Internet Files

CSIDL_LOCAL_APPDATA	The file-system directory that serves as a data repository for local, non-roaming applications. A typical path is C:\Users\username\AppData\Local.
CSIDL_MYDOCUMENTS	The virtual folder representing My Documents.A typical path is C:\Users\Username\Documents.
CSIDL_MYMUSIC	The file-system directory that serves as a common repository for music files. A typical path is C:\Users\Username\Music.
CSIDL_MYPICTURES	The file-system directory that serves as a common repository for image files. A typical path is C:\Users\Username\Pictures.
CSIDL_MYVIDEO	The file-system directory that serves as a common repository for video files. A typical path is C:\Users\Username\Videos.
CSIDL_NETHOOD	A file-system directory that contains the link objects that may exist in the My Network Places virtual folder. It is not the same as CSIDL_NETWORK, which represents the network namespace root. A typical path is C:\Users\Username\AppData\Roaming\Microsoft\Windows\Network Shortcuts.
CSIDL_NETWORK	A virtual folder representing My Network Places, the root of the network namespace hierarchy.
CSIDL_PERSONAL	The virtual folder representing the My Documents desktop item. This is equivalent to CSIDL_MYDOCUMENTS. A typical path is C:\Documents and Settings\username\My Documents.
CSIDL_PLAYLISTS	The virtual folder used to store play albums, typically C:\Users\username\My Music\Playlists.
CSIDL_PRINTERS	The virtual folder that contains installed printers.
CSIDL_PRINTHOOD	The file-system directory that contains the link objects that can exist in the Printers virtual folder. A typical path is C:\Users\username\AppData\Roaming\Microsoft\Windows\Printer Shortcuts.
CSIDL_PROFILE	The user's profile folder. A typical path is C:\Users\Username.
CSIDL_PROGRAMS	The file-system directory that contains the user's program groups, which are themselves file-system directories. A typical path is C:\Users\Username\AppData\Roaming\Microsoft\Windows\Start Menu\Programs.
CSIDL_RECENT	The file-system directory that contains shortcuts to the user's most recently used documents. A typical path is C:\Users\Username\AppData\Roaming\Microsoft\Windows\Recent.
CSIDL_SENDTO	The file-system directory that contains Send To menu items. A typical path is C:\Users\username\AppData\Roaming\Microsoft\Windows\SendTo.
CSIDL_STARTMENU	The file-system directory that contains Start menu items. A typical path in Windows XP is C:\Documents and Settings\username\Start Menu. A typical path in Windows Vista, Windows 7, or Windows 8 is C:\Users\Username\AppData\Roaming\Microsoft\Windows\Start Menu.
CSIDL_STARTUP	The file-system directory that corresponds to the user's Startup program group. A typical path is C:\Users\Username\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup.
CSIDL_TEMPLATES	The file-system directory that serves as a common repository for document templates. A typical path is C:\Users\username\AppData\Roaming\Microsoft\Windows\Templates.
HOMEPATH	Same as the standard environment variable.
TEMP	The temporary folder on the computer. A typical path is %USERPROFILE%\AppData\Local\Temp.

TMP	The temporary folder on the computer. A typical path is %USERPROFILE%\AppData\Local\Temp.
USERPROFILE	Same as CSIDL_PROFILE.
USERSID	Represents the current user-account security identifier (SID). For example, S-1-5-21-1714567821-1326601894-715345443-1026.

Reg File Creation

To create a scripting solution that changes a registry setting on a computer, you can use the .reg file in scripts to apply the registry setting to other computers.
A .reg file is a text file that contains data and paths for the Windows registry editor. Suppose you want add a a key MyKey with a ***String Place*** that contains the value "***New York***" and a ***dword MyKey*** with value ***00000001*** in the registry path ***HKEY_LOCAL_MACHINE\SOFTWARE***, here the syntax to use:

Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\MyKey\]

"Place "=" New York "

" MyKey"=dword:00000001

The text file must be saved with .reg extension, and to use it is enough open the file with regedit. You can also perform key remotion using the reg file. Suppose you want to create a script to remove the values created in the previous sample here the sintaxt to use:

Windows Registry Editor Version 5.00

[-HKEY_LOCAL_MACHINE\SOFTWARE\MyKey\]

The – indicates that the value must be removed.

Disable Data Execution Prevention

1) Open Command Prompt as admin and type

bcdedit.exe /set {current} nx AlwaysOff

2) Confirm and reboot

or

1) start regedit

2) go to *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows*

3) on *REG_DWORD LoadAppInit_DLLs* change value to 1 to enable AppInit mechanism if not enabled

4) on REG_DWORD RequireSignedAppInit - change value to 0

5) go to ***HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Windows***

6) create the new file ***REG_DWORD*** with name ***RequireSignedAppInit_DLLs - value 0***

7) close regedit

The AppInit_DLLs mechanism is disabled by default when secure boot is enabled.

If you want to use this method you must disable UEFI secure boot feature from the UEFI config

http://msdn.microsoft.com/en-us/library/windows/desktop/dd744762%28v=vs.85%29.aspx

Driver verifier

You can use Driver Verifier along with the ***!analyze*** debugger command to detect and display information about errors in your driver. When Driver Verifier detects an error, it generates a bug check. Then Windows breaks into the debugger and displays a brief description of the error.
From Elevated CMD:

verifier

This show Driver verifier configuration,

create new setting profile and will be asked you to restart.

To disable driver verifier input:

verifier / reset

WARNING
Driver Verifier may cause BSOD also with driver that "apparently works" preventing your system to start properly.

Critical Errors

A critical error (BSOD) can occur because of hardware problem a configuration issue, defective hardware, incompatible hardware or incompatible or corrupted driver.
The general suggestions to identify the cause are check the recently updated driver, or the new devices added to the pc and check hard disk, memory and OS integrity. The general suggestions to solve the issue are upgrade drivers and firmware to latest stable WHQL release. To resolve a disk corruption problem, check Event Viewer for error messages from SCSI, IDE, or other disk controllers in the system that might help pinpoint the device or driver that is causing the error. Perform a scandisk. Try disabling any virus scanners, backup programs, or disk defragmenter tools that continually monitor the system. You should also run hardware diagnostics supplied by the file system or the file system filter manufacturer.
To exclude problems related to memory, perform a memory integrity test. In case of issue try to remove dust and check the integrity again, if some errors are still detected replace the memory. After these step check the system files integrity using the instruction of the section Fix Damaged OS Files.
Sometimes BSOD could be caused by power supply problems or poor power management (especially in notebooks); make sure to check power supply is working correctly, and try to change power setting to maximum performance for the devices that could be involved with the issue.
If you have no idea of the possible cause enable Driver Verifier for any unsigned drivers. Remove them or disable them one by one until the erring driver is identified. You can find details about each type of critical error in the table below.

Critical Errors Table:

Code	Error Name	Description	Details
0x00000001	APC_INDEX_MISMATCH	This indicates that there has been a mismatch in the APC state index.	The most common cause of this bug check is when a file system or driver has a mismatched sequence of calls to disable and re-enable APCs. The key data item is the Thread->CombinedApcDisable field. The CombinedApcDisable field consists of two separate 16-bit fields: SpecialApcDisable and KernelApcDisable. A negative value of either field indicates that a driver has disabled special or normal APCs (respectively) without re-enabling them. A positive value indicates that a driver has enabled special or normal APCs too many times.
0x00000002	DEVICE_QUEUE_NOT_BUSY	Indicates that a device queue was expected to be busy, but was not.	See general suggestions above the table.
0x00000003	INVALID_AFFINITY_SET	It indicates a null of incorrect subset affinity.	See general suggestions above the table.
0x00000004	INVALID_DATA_ACCESS_TRAP	This error indicates an inability of the system to retrieve certain data that it needs to function.	See general suggestions above the table.

0x00000005	INVALID_PROCESS_ATTACH_ATTEMPT	This generally indicates that the thread was attached to a process in a situation where that is not allowed.	See general suggestions above the table.
0x00000006	INVALID_PROCESS_DETACH_ATTEMPT	It indicates a problem with an owned mutex or an unclean APC state.	See general suggestions above the table.
0x00000007	INVALID_SOFTWARE_INTERRUPT	It indicates a level not within the software range.	See general suggestions above the table.
0x00000008	IRQL_NOT_DISPATCH_LEVEL	It indicates an attempt to remove a device not at the dispatch level.	See general suggestions above the table.
0x00000009	IRQL_NOT_GREATER_OR_EQUAL	It indicates an IRQL that was expected to be greater or equal, but was not.	See general suggestions above the table.
0x0000000A	IRQL_NOT_LESS_OR_EQUAL	This indicates that Microsoft Windows or a kernel-mode driver accessed paged memory at DISPATCH_LEVEL or above.	This bug check is issued if paged memory (or invalid memory) is accessed when the IRQL is too high. The error that generates this bug check usually occurs after the installation of a faulty device driver, system service, or BIOS.
0x0000000B	NO_EXCEPTION_HANDLING_SUPPORT	It indicates that exception handling was not supported.	See general suggestions above the table.
0x0000000C	MAXIMUM_WAIT_OBJECTS_EXCEEDED	This indicates that the current thread exceeded the permitted number of wait objects.	This bug check results from the improper use of KeWaitForMultipleObjects or FsRtlCancellableWaitForMultipleObjects. The caller may pass a pointer to a buffer in this routine's WaitBlockArray parameter. The system will use this buffer to keep track of wait objects. If a buffer is supplied, the Count parameter may not exceed MAXIMUM_WAIT_OBJECTS. If no buffer is supplied, the Count parameter may not exceed THREAD_WAIT_OBJECTS.
0x0000000D	MUTEX_LEVEL_NUMBER_VIOLATION	It indicates an attempt to acquire a lower level mutex.	See general suggestions above the table.
0x0000000E	NO_USER_MODE_CONTEXT	It indicates an attempt to enter user mode with no context.	See general suggestions above the table.
0x0000000F	SPIN_LOCK_ALREADY_OWNED	This indicates that a request for a spin lock has been initiated when the spin lock was already owned.	See general suggestions above the table.
0x00000010	SPIN_LOCK_NOT_OWNED	This is a Windows character-mode Stop message. It indicates an attempt to release an unowned spin lock.	See general suggestions above the table.
0x00000011	THREAD_NOT_MUTEX_OWNER	It indicates an attempt to release a thread by a mutex non-owner.	See general suggestions above the table.
0x00000012	TRAP_CAUSE_UNKNOWN	This indicates that an unknown exception has occurred.	See general suggestions above the table.
0x00000013	EMPTY_THREAD_REAPER_LIST	It indicates the thread reaper list is corrupted (the reaper list was signaled, but no threads were present on the list).	See general suggestions above the table.
0x00000014	CREATE_DELETE_LOCK_NOT_LOCKED	It indicates an unsafe attempt to unlock a thread.	See general suggestions above the table.
0x00000015	LAST_CHANCE_CALLED_FROM_KMODE	This indicates that the last chance exception service was called from kernel mode.	See general suggestions above the table.
0x00000016	CID_HANDLE_CREATION	This indicates that a failure occurred creating a handle to represent a client ID.	See general suggestions above the table.
0x00000017	CID_HANDLE_DELETION	This indicates that a failure occurred deleting a handle to represent a client ID.	See general suggestions above the table.
0x00000018	REFERENCE_BY_POINTER	This indicates that the reference count of an object is illegal for the current state of the object.	Each time a driver uses a pointer to an object, the driver calls a kernel routine to increase the reference count of the object by one. When the driver is done with the pointer, the driver calls another kernel routine to decrease the reference count by 1. Drivers must match calls to the routines that increase (reference) and decrease (dereference) the reference count. This bug check is caused by an inconsistency in the object's reference count. Typically, the inconsistency is caused by a driver that decreases the reference count of an object too many times, making extra calls that dereference the object. This bug check can occur because an object's reference count goes to zero while there are still open handles to the object. It might also occur when the object's reference count drops below zero, whether or not there are open handles to the object.
0x00000019	BAD_POOL_HEADER	indicates that a pool header is corrupt	0x2 The special pool pattern check failed. 0x3 The pool freelist is corrupt. 0x5 A pair of adjacent pool entries have headers that contradict each other. At least one of them is corrupt. 0x6 The pool block header's previous size is too large. 0x7 The pool block header size is corrupt.

			0x8 The pool block header size is zero. 0x9 The pool block header size is corrupted (it is too large). 0xA The pool block header size is corrupt. 0xD, 0xE, 0xF, 0x23, 0x24, 0x25 The pool header of a freed block has been modified after it was freed. This is not typically the fault of the prior owner of the freed block; instead it is usually (but not always) due to the block preceding the freed block being overrun. 0x20 The pool block header size is corrupt. 0X21 The data following the pool block being freed is corrupt. 0X22 An address being freed does not have a tracking entry.
0x0000001A	MEMORY_MANAGEMENT	This indicates that a severe memory management error occurred.	The cause depends on the first parameter: 0x1 The fork clone block reference count is corrupt. 0x31 The image relocation fix-up table or code stream has been corrupted. This is probably a hardware error. 0x403 The page table and PFNs are out of sync . This is probably a hardware error, especially if parameters 3 & 4 differ by only a single bit. 0x411 A page table entry (PTE) has been corrupted. Parameter 2 is the address of the PTE. 0x777 The caller is unlocking a system cache address that is not currently locked. (This address was either never mapped or is being unlocked twice.) 0x778 The system is using the very last system cache view address, instead of preserving it. 0x780 0x781 The PTEs mapping the argument system cache view have been corrupted. 0x1000 A caller of MmGetSystemAddressForMdl* tried to map a fully-cached physical page as non-cached. This action would cause a conflicting hardware translation buffer entry, and so it was refused by the operating system. Since the caller specified "bug check on failure" in the requesting MDL, the system had no choice but to issue a bug check in this instance. 0x1010 The caller is unlocking a pageable section that is not currently locked. (This section was either never locked or is being unlocked twice.) 0x1233 A driver tried to map a physical memory page that was not locked. This is illegal because the contents or attributes of the page can change at any time. This is a bug in the code that made the mapping call. Parameter 2 is the page frame number of the physical page that the driver attempted to map 0x1234 The caller is trying lock a nonexistent pageable section. 0x1235 The caller is trying to protect an MDL with an invalid mapping. 0x3451 The PTEs of a kernel thread stack that has been swapped out are corrupted. 0x5003 The working set free list is corrupt. This is probably a hardware error. 0x5100 The allocation bitmap is corrupt. The memory manager is about to overwrite a virtual address that was already in use. 0x8884). Two pages on the standby list that were supposed to have identical page priority values do not, in fact, have identical page priority values. The differing values are captured in parameter 4. 0x8888 0x8889 Internal memory management structures are corrupted. 0x888A Internal memory management structures (likely the PTE or PFN) are corrupted. 0x41283 The working set index encoded in the PTE is corrupted. 0x41284 A PTE or the working set list is corrupted. 0x41286 The caller is trying to free an invalid pool address. 0x41785 The working set list is corrupted. 0x41287 An illegal page fault occurred while holding working set synchronization. Parameter 2 contains the referenced virtual address. 0x41790 A page table page has been corrupted. On a 64 bit version of Windows, parameter 2 contains the address of the PFN for the corrupted page table page. On

			a 32 bit version of Windows, parameter 2 contains a pointer to the number of used PTEs, and parameter 3 contains the number of used PTEs. 0x41792 A corrupted PTE has been detected. Parameter 2 contains the address of the PTE. Parameters 3/4 contain the low/high parts of the PTE. 0x61940 A PDE has been unexpectedly invalidated. 0x61946 The MDL being created is flawed. This almost always means the driver calling MmProbeAndLockPages is at fault. Typically the driver is attempting to create a Write MDL when it is being asked to process a paging Read. 0x03030303 The boot loader is broken. (This value applies only to Intel Itanium machines.) Other An unknown memory management error occurred.
0x0000001B	PFN_SHARE_COUNT	It indicates that a memory management page frame number (PFN) database element has a corrupted share count.	See general suggestions above the table.
0x0000001C	PFN_REFERENCE_COUNT	It indicates that a memory management page frame number (PFN) database element has a corrupted reference count.	See general suggestions above the table.
0x0000001D	NO_SPIN_LOCK_AVAILABLE	It indicates that no spin locks are available to allocate.	See general suggestions above the table.
0x0000001E	KMODE_EXCEPTION_NOT_HANDLED	This indicates that a kernel-mode program generated an exception which the error handler did not catch.	0x80000002: STATUS_DATATYPE_MISALIGNMENT An unaligned data reference was encountered. 0x80000003: STATUS_BREAKPOINT A breakpoint or ASSERT was encountered when no kernel debugger was attached to the system. 0xC0000005: STATUS_ACCESS_VIOLATION A memory access violation occurred. (Parameter 4 of the bug check is the address that the driver attempted to access.)
0x0000001F	SHARED_RESOURCE_CONV_ERROR	It indicates a shared resource conversion problem.	See general suggestions above the table.
0x00000020	KERNEL_APC_PENDING_DURING_EXIT	This indicates that an asynchronous procedure call (APC) was still pending when a thread exited.	The key data item is the APC disable count (Parameter 2) for the thread. If the count is nonzero, it will indicate the source of the problem. The APC disable count is decremented each time a driver calls KeEnterCriticalRegion, FsRtlEnterFileSystem, or acquires a mutex. The APC disable count is incremented each time a driver calls KeLeaveCriticalRegion, KeReleaseMutex, or FsRtlExitFileSystem. Because these calls should always be in pairs, the APC disable count should be zero when a thread exits. A negative value indicates that a driver has disabled APC calls without re-enabling them. A positive value indicates that the reverse is true. If you ever see this error, be very suspicious of all drivers installed on the machine -- especially unusual or non-standard drivers. This current IRQL (Parameter 3) should be zero. If it is not, the driver's cancellation routine may have caused this bug check by returning at an elevated IRQL. In this case, carefully note what was running (and what was closing) at the time of the crash, and note all of the installed drivers at the time of the crash. The cause in this case is usually a severe bug in a driver.
0x00000021	QUOTA_UNDERFLOW	This indicates that quota charges have been mishandled by returning more quota to a particular block than was previously charged.	See general suggestions above the table.
0x00000022	FILE_SYSTEM	This indicates a generic file system problem.	One possible cause of this bug check is disk corruption. Corruption in the file system or bad blocks (sectors) on the disk can induce this error. Corrupted SCSI and IDE drivers can also adversely affect the system's ability to read and write to the disk, thus causing the error.

			Another possible cause is depletion of nonpaged pool memory. If the nonpaged pool memory is completely depleted, this error can stop the system. However, during the indexing process, if the amount of available nonpaged pool memory is very low, another kernel-mode driver requiring nonpaged pool memory can also trigger this error.
0x00000023	FAT_FILE_SYSTEM	This indicates that a problem occurred in the FAT file system.	As previous.
0x00000024	NTFS_FILE_SYSTEM	This indicates a problem occurred in ntfs.sys, the driver file that allows the system to read and write to NTFS drives.	As previous.
0x00000025	NPFS_FILE_SYSTEM	This indicates that a problem occurred in the NPFS file system.	As previous.
0x00000026	CDFS_FILE_SYSTEM	This indicates that a problem occurred in the CD file system.	As previous.
0x00000027	RDR_FILE_SYSTEM	This indicates that a problem occurred in the SMB redirector file system.	As previous.
0x00000028	CORRUPT_ACCESS_TOKEN	It indicates that the security system encountered an invalid access token.	See general suggestions above the table.
0x00000029	SECURITY_SYSTEM	It indicates a problem internal to the security system.	See general suggestions above the table.
0x0000002A	INCONSISTENT_IRP	This indicates that an IRP was found to contain inconsistent information.	An IRP was discovered to be in an inconsistent state. Usually this means some field of the IRP was inconsistent with the remaining state of the IRP. An example would be an IRP that was being completed, but was still marked as being queued to a driver's device queue.
0x0000002B	PANIC_STACK_SWITCH	This indicates that the kernel mode stack was overrun.	This error normally appears when a kernel-mode driver uses too much stack space. It can also appear when serious data corruption occurs in the kernel.
0x0000002C	PORT_DRIVER_INTERNAL	It indicates an internal error in a port driver.	See general suggestions above the table.
0x0000002D	SCSI_DISK_DRIVER_INTERNAL	It indicates an internal error in a SCSI hard disk driver.	See general suggestions above the table.
0x0000002E	DATA_BUS_ERROR	This typically indicates that a parity error in system memory has been detected.	This error is almost always caused by a hardware problem a configuration issue, defective hardware, or incompatible hardware. The most common hardware problems that can cause this error are defective RAM, Level 2 (L2) RAM cache errors, or video RAM errors. Hard disk corruption can also cause this error. This bug check can also be caused when a device driver attempts to access an address in the 0x8xxxxxxx range that does not exist (in other words that does not have a physical address mapping).
0x0000002F	INSTRUCTION_BUS_ERROR	It indicates an instruction bus error.	This error is almost always caused by a hardware problem a configuration issue, defective hardware, or incompatible hardware. The most common hardware problems that can cause this error are defective RAM, video RAM or motherboard. Hard disk corruption can also cause this error.
0x00000030	SET_OF_INVALID_CONTEXT	This indicates that the stack pointer in a trap frame had an invalid value.	This bug check occurs when some routine attempts to set the stack pointer in the trap frame to a lower value than the current stack pointer value. If this error were not caught, it would cause the kernel to run with a stack pointer pointing to stack which is no longer valid.
0x00000031	PHASE0_INITIALIZATION_FAILED	This indicates that system initialization failed.	System initialization failed at a very early stage.
0x00000032	PHASE1_INITIALIZATION_FAILED	This indicates that system initialization failed.	See general suggestions above the table.
0x00000033	UNEXPECTED_INITIALIZATION_CALL	This indicates an unexpected call during inizalization.	Initialization of the Windows Executive failed during phase 1 (during phase 4 of system startup). There might be a problem with a device driver.
0x00000034	CACHE_MANAGER	This indicates that a problem occurred in the file system's cache manager.	One possible cause of this bug check is depletion of nonpaged pool memory. If the nonpaged pool memory is completely depleted, this error can stop the system. However, during the indexing process, if the amount of available nonpaged pool memory is very low, another kernel-mode driver requiring nonpaged pool memory can also trigger this error.
0x00000035	NO_MORE_IRP_STACK_LOCATIONS	This bug check occurs when the IoCallDriver packet has no more stack locations remaining.	A higher-level driver has attempted to call a lower-level driver through the IoCallDriver interface, but there are

			no more stack locations in the packet. This will prevent the lower-level driver from accessing its parameters. This is a disastrous situation, since the higher level driver is proceeding as if it has filled in the parameters for the lower level driver (as required). But since there is no stack location for the latter driver, the former has actually written off the end of the packet. This means that some other memory has been corrupted as well.
0x00000036	DEVICE_REFERENCE_COUNT_NOT_ZERO	This indicates that a driver attempted to delete a device object that still had a positive reference count.	A device driver has attempted to delete one of its device objects from the system, but the reference count for that object was non-zero. This means there are still outstanding references to the device. (The reference count indicates the number of reasons why this device object cannot be deleted.) This is a bug in the calling device driver.
0x00000037	FLOPPY_INTERNAL_ERROR	It indicates a floppy disk driver internal error.	See general suggestions above the table.
0x00000038	SERIAL_DRIVER_INTERNAL	It indicates a serial device driver internal error.	See general suggestions above the table.
0x00000039	SYSTEM_EXIT_OWNED_MUTEX	This indicates that the worker routine returned without releasing the mutex object that it owned.	The worker routine returned while it still owned a mutex object. The current worker thread will proceed to run other unrelated work items, and the mutex will never be released.
0x0000003A	SYSTEM_UNWIND_PREVIOUS_USER	It indicates an attempt was made to unwind through the system service dispatcher into user mode.	See general suggestions above the table.
0x0000003B	SYSTEM_SERVICE_EXCEPTION	This indicates that an exception happened while executing a routine that transitions from non-privileged code to privileged code.	This error has been linked to excessive paged pool usage and may occur due to user-mode graphics drivers crossing over and passing bad data to the kernel code.
0x0000003C	INTERRUPT_UNWIND_ATTEMPTED	It indicates an unwind operation was initiated in an interrupt service routine that attempted to unwind through the interrupt dispatcher.	See general suggestions above the table.
0x0000003D	INTERRUPT_EXCEPTION_NOT_HANDLED	It indicates an exception was raised in an interrupt service routine which was not handled by the interrupt service routine.	See general suggestions above the table.
0x0000003E	MULTIPROCESSOR_CONFIGURATION_NOT_SUPPORTED	This indicates that the system has multiple processors, but they are asymmetric in relation to one another.	In order to be symmetric, all processors must be of the same type and level. This system contains processors of different types.
0x0000003F	NO_MORE_SYSTEM_PTES	This is the result of a system which has performed too many I/O actions. This has resulted in fragmented system page table entries (PTE).	In almost all cases, the system is not actually out of PTEs. Rather, a driver has requested a large block of memory, but there is no contiguous block of sufficient size to satisfy this request. Often video drivers will allocate large amounts of kernel memory that must succeed. Some backup programs do the same.
0x00000040	TARGET_MDL_TOO_SMALL	This indicates that a driver has improperly used IoBuildPartialMdl.	This is a driver bug. A driver has called the IoBuildPartialMdl function and passed it an MDL to map part of a source MDL, but the target MDL is not large enough to map the entire range of addresses requested.
0x00000041	MUST_SUCCEED_POOL_EMPTY	This indicates that a kernel-mode thread has requested too much must-succeed pool.	In Microsoft Windows 2000, only a small amount of must-succeed pool is permitted. In later 2k based OS, no driver is permitted to request must-succeed pool. If a must-succeed request cannot be filled, this bug check is issued.
0x00000042	ATDISK_DRIVER_INTERNAL	It indicates a hard disk device driver internal error.	See general suggestions above the table.
0x00000043	NO_SUCH_PARTITION	It indicates an attempt to access a missing partition.	The partition on the hard drive is missing or damaged or the hard drive is faulty.
0x00000044	MULTIPLE_IRP_COMPLETE_REQUESTS	This indicates that a driver has tried to request an IRP be completed that is already complete.	A driver has called IoCompleteRequest to ask that an IRP be completed, but the packet has already been completed. This is a tough bug to find because the simplest case a driver that attempted to complete its own packet twice is usually not the source of the problem. More likely, two separate drivers each believe that they own the packet, and each has attempted to complete it. The first request succeeds, and the second fails, resulting in this bug check. Tracking down which drivers in the system caused the error is difficult, because the trail of the first driver has been covered by the second. However, the driver stack for the current request can be found by examining the device object fields in each of the stack locations.

0x00000045	INSUFFICIENT_SYSTEM_MAP_REGS	It indicates that an attempt was made to allocate more map registers than are allocated to an adapter.	See general suggestions above the table.
0x00000046	DEREF_UNKNOWN_LOGON_SESSION	It indicates that a token was deleted that was not part of any known logon session.	See general suggestions above the table.
0x00000047	REF_UNKNOWN_LOGON_SESSION	It indicates that a token was created that was not part of any known logon session.	See general suggestions above the table.
0x00000048	CANCEL_STATE_IN_COMPLETED_IRP	This indicates that an I/O request packet (IRP) was completed, and then was subsequently canceled.	An IRP that had a Cancel routine set was completed normally, without cancellation. But after it was complete, a driver called the IRP's Cancel routine. This could be caused by a driver that completed the IRP and then attempted to cancel it. It could also be caused by two drivers each trying to access the same IRP in an improper way. The cancel routine parameter can be used to determine which driver or stack caused the bug check.
0x00000049	PAGE_FAULT_WITH_INTERRUPTS_OFF	It indicates that a page fault occurred while interrupts were disabled.	See general suggestions above the table.
0x0000004A	IRQL_GT_ZERO_AT_SYSTEM_SERVICE	This indicates that a thread is returning to user mode from a system call when its IRQL is still above PASSIVE_LEVEL.	See general suggestions above the table.
0x0000004B	STREAMS_INTERNAL_ERROR	This indicates an internal error in the Streams environment or in a Streams driver.	See general suggestions above the table.
0x0000004C	FATAL_UNHANDLED_HARD_ERROR	This indicates a fatal hard error (STATUS error) occurred before the hard error handler was available. There are several reasons why this error might occur: a Registry hive file could not be loaded because it is either corrupted or missing; Winlogon or Windows unexpectedly did not start; or a driver or system DLL is corrupted.	See general suggestions above the table.
0x0000004D	NO_PAGES_AVAILABLE	This indicates that no free pages are available to continue operations.	To see general memory statistics, use the !vm 3 extension. This bug check can occur for any of the following reasons: A driver has blocked, deadlocking the modified or mapped page writers. Examples of this include mutex deadlocks or accesses to paged out memory in file system drivers or filter drivers. This indicates a driver bug. If Parameter 1 or Parameter 2 is large, then this is a possibility. Use !vm 3. A storage driver is not processing requests. Examples of this are stranded queues and non-responding drives. This indicates a driver bug. If Parameter 1 or Parameter 2 is large, then this is a possibility. Use !vm 8, followed by !process 0 7. A high-priority realtime thread has starved the balance set manager from trimming pages from the working set, or starved the modified page writer from writing them out. This indicates a bug in the component that created this thread. This situation is difficult to analyze. Try using !ready. Try also !process 0 7 to list all threads and see if any have accumulated excessive kernel time as well as what their current priorities are. Such processes may have blocked out the memory management threads from making pages available. Windows XP and Windows 2000: Not enough pool is available for the storage stack to write out modified pages. This indicates a driver bug. If Parameter 3 is small, then this is a possibility. Use !vm and !poolused 2. Windows 2000: All the processes have been trimmed to their minimums and all modified pages written, but still no memory is available. The freed memory must be stuck in transition pages with non-zero reference counts -- thus they cannot be put on the freelist. A driver is neglecting to unlock the pages preventing the reference counts from going to zero which would free

			the pages. This may be due to transfers that never finish, causing the driver routines to run endlessly, or to other driver bugs. If Parameter 4 is large, then this is a possibility. But it is very hard to find the driver. Try the !process 0 1 extension and look for any drivers that have a lot of locked pages. If the problem cannot be found, then try booting with a kernel debugger attached from the beginning, and monitor the situation.
0x0000004E	PFN_LIST_CORRUPT	This indicates that the page frame number (PFN) list is corrupted.	This error is typically caused by a driver passing a bad memory descriptor list. For example, the driver might have called MmUnlockPages twice with the same list. The following parameters are displayed on the blue screen. Parameter 1 indicates the type of violation. The cause of error and meaning of the other parameters depends on the value of Parameter 1. 0x01 The list head was corrupted. 0x02 A list entry was corrupted. 0x07 A driver has unlocked a certain page more times than it locked it. 0x8D The page-free list is corrupted. This error code most likely indicates a hardware issue. 0x8F The free or zeroed page listhead is corrupted. 0x99 A page table entry (PTE) or PFN is corrupted. 0x9A A driver attempted to free a page that is still locked for IO.
0x0000004F	NDIS_INTERNAL_ERROR	It indicates an internal error in the NDIS wrapper or an NDIS driver.	See general suggestions above the table.
0x00000050	PAGE_FAULT_IN_NONPAGED_AREA	This indicates that invalid system memory has been referenced.	Bug check 0x50 usually occurs after the installation of faulty hardware or in the event of failure of installed hardware (usually related to defective RAM, be it main memory, L2 RAM cache, or video RAM). Another common cause is the installation of a faulty system service. Antivirus software can also trigger this error, as can a corrupted NTFS volume.
0x00000051	REGISTRY_ERROR	This indicates that a severe registry error has occurred.	Something has gone wrong with the registry. If a kernel debugger is available, get a stack trace. This error may indicate that the registry encountered an I/O error while trying to read one of its files. This can be caused by hardware problems or file system corruption. It may also occur due to a failure in a refresh operation, which is used only in by the security system, and then only when resource limits are encountered.
0x00000052	MAILSLOT_FILE_SYSTEM	It indicates a mailslot file system problem.	See general suggestions above the table.
0x00000053	NO_BOOT_DEVICE	It indicates that no boot driver was successfully initialized.	It may also occur due to a failure in hard drive or due to a wrong boot configuration or system installation.
0x00000054	LM_SERVER_INTERNAL_ERROR	It indicates an internal error in the Windows Server.	See general suggestions above the table.
0x00000055	DATA_COHERENCY_EXCEPTION	It indicates an inconsistency between pages in the primary and secondary data caches.	It indicates an inconsistency between pages in the primary and secondary instruction
0x00000056	INSTRUCTION_COHERENCY_EXCEPTION	It indicates an inconsistency between pages in the primary and secondary instruction caches.	See general suggestions above the table.
0x00000057	XNS_INTERNAL_ERROR	It indicates an XNS internal error. You might need to replace your network card.	See general suggestions above the table.
0x00000058	FTDISK_INTERNAL_ERROR	This is issued if the system is booted from the wrong copy of a mirrored partition.	The hives are indicating that the mirror is valid, but it is not. The hives should actually be pointing to the shadow partition. This is almost always caused by the primary partition being revived. Reboot the system from the shadow partition.
0x00000059	PINBALL_FILE_SYSTEM	This indicates that a problem occurred in the Pinball file system.	One possible cause of this bug check is depletion of nonpaged pool memory. If the nonpaged pool memory is completely depleted, this error can stop the system. However, during the indexing process, if the amount of

			available nonpaged pool memory is very low, another kernel-mode driver requiring nonpaged pool memory can also trigger this error. To resolve a nonpaged pool memory depletion problem: Add new physical memory to the computer. This will increase the quantity of nonpaged pool memory available to the kernel.
0x0000005A	CRITICAL_SERVICE_FAILED	It indicates that a critical service failed to initialize while starting the LastKnownGood	See general suggestions above the table.
0x0000005B	SET_ENV_VAR_FAILED	It indicates a critical service failed to initialize, but the LastKnownGood environment variable could not be set.	See general suggestions above the table.
0x0000005C	HAL_INITIALIZATION_FAILED	This indicates that the Hardware Abstraction Layer initialization has failed.	See general suggestions above the table.
0x0000005D	UNSUPPORTED_PROCESSOR	This indicates that the computer is attempting to run Windows on an unsupported processor.	Windows requires a higher-grade processor than the one you are using or with a different architecture.
0x0000005E	OBJECT_INITIALIZATION_FAILED	Phase 0 initialization of the object manager failed.	This can only happen during the relatively short period of time that the Windows Executive is being initialized, during phase 4 of Windows startup. This might be a hardware problem or a severe driver issue.
0x0000005F	SECURITY_INITIALIZATION_FAILED	Phase 0 security initialization failed.	As above.
0x00000060	PROCESS_INITIALIZATION_FAILED	Phase 0 process initialization failed.	As above.
0x00000061	HAL1_INITIALIZATION_FAILED	Phase 1 initialization of the Hardware Abstraction Layer (HAL) failed.	As above.
0x00000062	OBJECT1_INITIALIZATION_FAILED	Phase 1 initialization of the object manager failed.	As above.
0x00000063	SECURITY1_INITIALIZATION_FAILED	Phase 1 security initialization failed.	As above.
0x00000064	SYMBOLIC_INITIALIZATION_FAILED	Symbolic link initialization failed.	As above.
0x00000065	MEMORY1_INITIALIZATION_FAILED	Phase 1 memory initialization failed.	As above.
0x00000066	CACHE_INITIALIZATION_FAILED	Cache initialization failed.	As above.
0x00000067	CONFIG_INITIALIZATION_FAILED	This bug check indicates that the registry configuration failed.	The registry could not allocate the pool that it needed to contain the registry files. This situation should never occur, because the register allocates this pool early enough in system initialization so that plenty of paged pool should be available. This can only happen during the relatively short period of time that the Windows.
0x00000068	FILE_INITIALIZATION_FAILED	File system initialization failed.	Caused by corrupted filesystem or faulty hard driver or related drivers.
0x00000069	IO1_INITIALIZATION_FAILED	This bug check indicates that the initialization of the I/O system failed for some reason.	Most likely, the setup routine has improperly installed the system, or a user has reconfigured the system or there might be a problem with a device driver.
0x0000006A	LPC_INITIALIZATION_FAILED	Local procedure call (LPC) initialization failed.	This can only happen during the relatively short period of time that the Windows Executive is being initialized, during phase 4 of Windows startup. This might be a hardware problem or a severe driver issue.
0x0000006B	PROCESS1_INITIALIZATION_FAILED	Phase 1 process initialization failed.	As above.
0x0000006C	REFMON_INITIALIZATION_FAILED	Reference monitor initialization failed.	As above.
0x0000006D	SESSION1_INITIALIZATION_FAILED	Session manager virtual memory allocation parameters failed during session manager initialization.	As above.
0x0000006E	SESSION2_INITIALIZATION_FAILED	The session manager virtual memory environment failed to initialize.	As above.
0x0000006F	SESSION3_INITIALIZATION_FAILED	Session manager process creation failed.	As above.
0x00000070	SESSION4_INITIALIZATION_FAILED	During session manager initialization, a resume thread operation failed.	As above.
0x00000071	SESSION5_INITIALIZATION_FAILED	During phase 1 initialization of the Windows Executive, the session manager terminated.	As above.
0x00000072	ASSIGN_DRIVE_LETTERS_FAILED	It indicates that a drive letter assignment failed.	Caused by corrupted filesystem or faulty storage device or related drivers.
0x00000073	CONFIG_LIST_FAILED	This bug check indicates that one of the top-level registry keys, also known as core system hives, cannot be linked in the registry tree.	The registry hive that cannot be linked might be SAM, SECURITY, SOFTWARE, or DEFAULT. The hive is valid, because it was loaded successfully. Examine Parameter 2 to see why the hive could not be linked in the registry tree. One common cause of this error is that the Windows operating system is out of disk space on the system drive. (In this situation, this parameter is 0xC000017D, STATUS_NO_LOG_SPACE.) Another common problem is that an attempt to allocate pool has failed. (In this situation, Parameter 2 is 0xC000009A, STATUS_INSUFFICIENT_RESOURCES.) You must investigate other status codes.
0x00000074	BAD_SYSTEM_CONFIG_INFO	This bug check indicates that there is an error in the registry.	The BAD_SYSTEM_CONFIG_INFO bug check occurs if the SYSTEM hive is corrupt. However, this corruption is unlikely, because the boot loader, known as NT Loader (NTLDR) in versions of Windows prior to Vista, checks a hive for corruption when it loads the hive. This bug check can also occur if some critical registry keys and values are

			missing. Thee keys and values might be missing if a user manually edited the registry. Try restarting the computer by selecting "last known good configuration" in the boot options. If the restart does not fix the problem, the registry damage is too extensive. You must reinstall the OS or use the Emergency Repair Disk (ERD) that you previously created by using the Windows Backup tool.
0x00000075	CANNOT_WRITE_CONFIGURATION	This bug check indicates that the SYSTEM registry hive file cannot be converted to a mapped file.	The CANNOT_WRITE_CONFIGURATION bug check typically occurs if the system is out of pool and the Windows operating system cannot reopen the hive. This bug check should almost never occur, because the conversion of the hive file occurs early enough during system initialization so that enough pool should be available.
0x00000076	PROCESS_HAS_LOCKED_PAGES	This bug check indicates that a driver failed to release locked pages after an I/O operation, or that it attempted to unlock pages that were already unlocked.	The driver either failed to unlock pages that it locked (parameter 1 value is 0x0), or the driver is attempting to unlock pages that have not been locked or that have already been unlocked (parameter 1 value is 0x1). If the parameter 1 value is 0x0 First use the !search extension on the current process pointer throughout all of physical memory. This extension might find at least one memory descriptor list (MDL) that points to the current process. Next, use !search on each MDL that you find to obtain the I/O request packet (IRP) that points to the current process. From this IRP, you can identify which driver is leaking the pages. Otherwise, you can detect which driver caused the error by editing the registry: -In the \\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management registry key, create or edit the TrackLockedPages value, and then set it equal to DWORD 1. -Restart the computer. The system then saves stack traces, so you can easily identify the driver that caused the problem. If the driver causes the same error again, bug check 0xCB (DRIVER_LEFT_LOCKED_PAGES_IN_PROCESS) is issued, and the name of the driver that causes this error is displayed on the blue screen and stored in memory at the location (PUNICODE_STRING) KiBugCheckDriver. If the parameter 1 value is 0x1 Examine the driver source code that locks and unlocks memory, and try to locate an instance where memory is unlocked without first being locked.
0x00000077	KERNEL_STACK_INPAGE_ERROR	This bug check indicates that the requested page of kernel data from the paging file could not be read into memory.	If the first parameter is 0 or 1, the stack signature in the kernel stack was not found. This error is probably caused by defective hardware, such as a RAM error. If the first parameter is 2, the driver stack returned an inconsistent status for the read of the page. For example, the driver stack returned a success status even though it did not read the whole page. If the first parameter is any value other than 0, 1, or 2, the value of the first parameter is an NTSTATUS error code that the driver stack returns after it tries to retrieve the page of kernel data. You can determine the exact cause of this error from the I/O status code (the second parameter). Some common status codes include the following: 0xC000009A, or STATUS_INSUFFICIENT_RESOURCES, indicates a lack of nonpaged pool resources. This status code indicates a driver error in the storage stack. (The storage stack should always be able to retrieve this data, regardless of software resource availability.) 0xC000009C, or STATUS_DEVICE_DATA_ERROR, indicates bad blocks (sectors) on the hard disk. 0xC000009D, or STATUS_DEVICE_NOT_CONNECTED, indicates defective or loose cabling, termination, or that the controller does not see the hard disk drive. 0xC000016A, or STATUS_DISK_OPERATION_FAILED, indicates bad blocks (sectors) on the hard disk.

			0xC0000185, or STATUS_IO_DEVICE_ERROR, indicates improper termination or defective cabling on SCSI devices or that two devices are trying to use the same IRQ. These status codes are the most common ones that have specific causes. For more information about other possible status codes that might be returned, see the Ntstatus.h file in the Microsoft Windows Driver Kit (WDK). A virus infection can also cause this bug check.
0x00000078	PHASE0_EXCEPTION	This bug check occurs when an unexpected break is encountered during HAL initialization.	This break can occur if you have set the /break parameter in your boot settings but have not enabled kernel debugging.
0x00000079	MISMATCHED_HAL	This bug check indicates that the Hardware Abstraction Layer (HAL) revision level or configuration does not match that of the kernel or the computer.	The MISMATCHED_HAL bug check often occurs when a user manually updates Ntoskrnl.exe or Hal.dll. The error can also indicate that one of those two files is out of date or the computer might erroneously have a multiprocessor HAL and a single-processor kernel installed, or vice versa. The Ntoskrnl.exe kernel file is for single-processor systems and Ntkrnlmp.exe is for multiprocessor systems. However, these file names correspond to the files on the installation media. After you have installed the Windows operating system, the file is renamed to Ntoskrnl.exe, regardless of the source file that is used. The HAL file also uses the name Hal.dll after installation, but there are several possible HAL files on the installation media. For more information, see "Installing the Checked Build" in the Windows Driver Kit (WDK).
0x0000007A	KERNEL_DATA_INPAGE_ERROR	This bug check indicates that the requested page of kernel data from the paging file could not be read into memory.	Frequently, you can determine the cause of the KERNEL_DATA_INPAGE_ERROR bug check from the error status (Parameter 2). Some common status codes include the following: 0xC000009A, or STATUS_INSUFFICIENT_RESOURCES, indicates a lack of nonpaged pool resources. 0xC000009C, or STATUS_DEVICE_DATA_ERROR, typically indicates bad blocks (sectors) on the hard disk. 0xC000009D, or STATUS_DEVICE_NOT_CONNECTED, indicates defective or loose cabling, termination, or that the controller does not see the hard disk. 0xC000016A, or STATUS_DISK_OPERATION_FAILED, indicates bad blocks (sectors) on the hard disk. 0xC0000185, or STATUS_IO_DEVICE_ERROR, indicates improper termination or defective cabling on SCSI devices or that two devices are trying to use the same IRQ. 0xC000000E, or STATUS_NO_SUCH_DEVICE, indicates a hardware failure or an incorrect drive configuration. Check your cables and check the drive with the diagnostic utility available from your drive manufacturer. If you are using older PATA (IDE) drives, this status code can indicate an incorrect master/subordinate drive configuration. These status codes are the most common ones that have specific causes. For more information about other possible status codes that can be returned, see the Ntstatus.h file in the Microsoft Windows Driver Kit (WDK). Another common cause of this error message is defective hardware or failing RAM. A virus infection can also cause this bug check.
0x0000007B	INACCESSIBLE_BOOT_DEVICE	This bug check indicates that the Microsoft Windows operating system has lost access to the system partition during startup.	The INACCESSIBLE_BOOT_DEVICE bug check frequently occurs because of a boot device failure. During I/O system initialization, the boot device driver might have failed to initialize the boot device (typically a hard disk). File system initialization might have failed because it did not recognize the data on the boot device. Also, repartitioning the system partition or installing a new SCSI adapter or disk controller might induce this error. This error can also occur because of incompatible disk hardware. If the error occurred at the initial setup of the system, the system might have been installed on an unsupported disk or SCSI controller. Some controllers are supported only by drivers that are in the Windows Driver Library (WDL). (These drivers require the user to do a custom installation.)
0x0000007C	BUGCODE_NDIS_DRIVER	This bug check indicates that a problem occurred with an NDIS driver.	Parameter 1 indicates the specific cause of the BUGCODE_NDIS_DRIVER bug check.

			If one of the bug check parameters specifies the address of the miniport block, you can obtain more information by using !ndiskd.miniport together with this address. If one of the bug check parameters specifies the address of the packet descriptor that the driver uses, you can obtain more information by using !ndiskd.pkt together with this address.
0x0000007D	INSTALL_MORE_MEMORY	This bug check indicates that there is not enough memory to start up the Microsoft Windows operating system.	The Windows operating system does not have sufficient memory to complete the startup process.
0x0000007E	SYSTEM_THREAD_EXCEPTION_NOT_HANDLED	This bug check indicates that a system thread generated an exception that the error handler did not catch.	The SYSTEM_THREAD_EXCEPTION_NOT_HANDLED bug check is a very common bug check. To interpret it, you must identify which exception was generated. Common exception codes include the following: 0x80000002: STATUS_DATATYPE_MISALIGNMENT indicates an unaligned data reference was encountered. 0x80000003: STATUS_BREAKPOINT indicates a breakpoint or ASSERT was encountered when no kernel debugger was attached to the system. 0xC0000005: STATUS_ACCESS_VIOLATION indicates a memory access violation occurred. For a complete list of exception codes, see the Ntstatus.h file that is located in the inc directory of the Microsoft Windows Driver Kit (WDK). If you are not equipped to debug this problem, you should use some basic troubleshooting techniques. Make sure you have enough disk space. If a driver is identified in the bug check message, disable the driver or check with the manufacturer for driver updates. Try changing video adapters. Check with your hardware vendor for any BIOS updates. Disable BIOS memory options such as caching or shadowing. If you plan to debug this problem, you might find it difficult to obtain a stack trace. Parameter 2 (the exception address) should identify the driver or function that caused this problem. If exception code 0x80000003 occurs, a hard-coded breakpoint or assertion was hit, but the system was started with the /NODEBUG switch. This problem should rarely occur. If it occurs repeatedly, make sure that a kernel debugger is connected and the system is started with the /DEBUG switch. If exception code 0x80000002 occurs, the trap frame supplies additional information. If you do not know the specific cause of the exception, consider the following issues: Hardware incompatibility. Make sure that any new hardware that is installed is compatible with the installed version of Windows. Faulty device driver or system service. A faulty device driver or system service might be responsible for this error. Hardware issues, such as BIOS incompatibilities, memory conflicts, and IRQ conflicts can also generate this error. If a driver is listed by name within the bug check message, disable or remove that driver. Disable or remove any drivers or services that were recently added. If the error occurs during the startup sequence and the system partition is formatted with NTFS file system, you might be able to use Safe Mode to rename or delete the faulty driver. If the driver is used as part of the system startup process in Safe Mode, you must start the computer by using the Recovery Console to access the file.

			If the problem is associated with Win32k.sys, the source of the error might be a third-party remote control program. If such software is installed, you can remove the service by starting the computer by using the Recovery Console and then deleting the offending system service file. Check the System Log in Event Viewer for additional error messages that might help identify the device or driver that is causing bug check 0x7E. You can also disable memory caching of the BIOS might to try to resolve the error. You should also run hardware diagnostics, especially the memory scanner, that the system manufacturer supplies. For more information about these procedures, see the owner's manual for your computer. The error that generates this message can occur after the first restart during Windows Setup, or after Setup is finished. A possible cause of the error is lack of disk space for installation and system BIOS incompatibilities. For problems during Windows installation that are associated with lack of disk space, reduce the number of files on the target hard disk drive. Check for and delete any temporary files that you do not have to have, Internet cache files, application backup files, and .chk files that contain saved file fragments from disk scans. You can also use another hard disk drive with more free space for the installation. You can resolve BIOS problems by upgrading the system BIOS version.
0x0000007F	UNEXPECTED_KERNEL_MODE_TRAP	This bug check indicates that the Intel CPU generated a trap and the kernel failed to catch this trap.	This trap could be a bound trap (a trap the kernel is not permitted to catch) or a double fault (a fault that occurred while processing an earlier fault, which always results in a system failure). Bug check 0x7F typically occurs after you install a faulty or mismatched hardware (especially memory) or if installed hardware fails. A double fault can occur when the kernel stack overflows. This overflow occurs if multiple drivers are attached to the same stack. For example, if two file system filter drivers are attached to the same stack and then the file system recurses back in, the stack overflows.
0x00000080	NMI_HARDWARE_FAILURE	This bug check indicates that a hardware malfunction has occurred.	A variety of hardware malfunctions can cause the NMI_HARDWARE_FAILURE bug check. The exact cause is difficult to determine.
0x00000081	SPIN_LOCK_INIT_FAILURE	It indicates the system was no able to lock hard drive spin.	See general suggestions above the table.
0x00000082	DFS_FILE_SYSTEM	It indicates a Distributed file system (Dfs) problem.	See general suggestions above the table.
0x00000085	SETUP_FAILURE	This bug check indicates that a fatal error occurred during setup.	See general suggestions above the table.
0x0000008B	MBR_CHECKSUM_MISMATCH	This bug check indicates that a mismatch has occurred in the MBR checksum.	The MBR_CHECKSUM_MISMATCH bug check occurs during the boot process when the MBR checksum that the Microsoft Windows operating system calculates does not match the checksum that the loader passes in. This error typically indicates a virus or an hard drive problem.
0x0000008E	KERNEL_MODE_EXCEPTION_NOT_HANDLED	This bug check indicates that a kernel-mode application generated an exception that the error handler did not catch.	The KERNEL_MODE_EXCEPTION_NOT_HANDLED bug check is a very common bug check. To interpret it, you must identify which exception was generated. Common exception codes include the following: 0x80000002: STATUS_DATATYPE_MISALIGNMENT indicates that an unaligned data reference was encountered. 0x80000003: STATUS_BREAKPOINT indicates that a breakpoint or ASSERT was encountered when no kernel debugger was attached to the system. 0xC0000005: STATUS_ACCESS_VIOLATION indicates that a memory access violation occurred. For a complete list of exception codes, see the Ntstatus.h file that is located in the inc directory of the Microsoft Windows Driver Kit (WDK).
0x0000008F	PP0_INITIALIZATION_FAILED	This bug check indicates that the Plug and Play (PnP) manager could not be initialized.	An error occurred during Phase 0 initialization of the kernel-mode PnP manager.
0x00000090	PP1_INITIALIZATION_FAILED	This bug check indicates that the Plug and Play (PnP) manager could not be initialized.	An error occurred during Phase 1 initialization of the kernel-mode PnP manager. Phase 1 is where most of the initialization is done, including setting up the registry files and other

			environment settings for drivers to call during the subsequent I/O initialization.
0x00000092	UP_DRIVER_ON_MP_SYSTEM	This bug check indicates that a uniprocessor-only driver has been loaded on a multiprocessor system.	A driver that is compiled to work only on uniprocessor machines has been loaded, but the Microsoft Windows operating system is running on a multiprocessor system with more than one active processor.
0x00000093	INVALID_KERNEL_HANDLE	This bug check indicates that an invalid or protected handle was passed to NtClose.	The INVALID_KERNEL_HANDLE bug check indicates that some kernel code (for example, a server, redirector, or another driver) tried to close an invalid handle or a protected handle.
0x00000094	KERNEL_STACK_LOCKED_AT_EXIT	This bug check indicates that a thread exited while its kernel stack was marked as not swappable	See general suggestions above the table.
0x00000096	INVALID_WORK_QUEUE_ITEM	This bug check indicates that a queue entry was removed that contained a NULL pointer.	The INVALID_WORK_QUEUE_ITEM bug check occurs when KeRemoveQueue removes a queue entry whose flink or blink field is NULL. Any queue misuse can cause this error. But typically this error occurs because worker thread work items are misused. An entry on a queue can be inserted on the list only one time. When an item is removed from a queue, its flink field is set to NULL. Then, when this item is removed the second time, this bug check occurs. In most situations, the queue that is being referenced is an ExWorkerQueue (executive worker queue). To help identify the driver that caused the error, Parameter 4 displays the address of the worker routine that would have been called if this work item had been valid. However, if the queue that is being referenced is not an ExWorkerQueue, this parameter is not useful.
0x00000097	BOUND_IMAGE_UNSUPPORTED	It indicates that a reference to an image that is stored in a database is not supported.	See general suggestions above the table.
0x00000098	END_OF_NT_EVALUATION_PERIOD	This bug check indicates that the trial period for the Microsoft Windows operating system has ended.	Your installation of the Windows operating system is an evaluation unit with an expiration date. The trial period is over.
0x00000099	INVALID_REGION_OR_SEGMENT	This bug check indicates that ExInitializeRegion or ExInterlockedExtendRegion was called with an invalid set of parameters.	See general suggestions above the table.
0x0000009A	SYSTEM_LICENSE_VIOLATION	This bug check indicates that the software license agreement has been violated.	The Microsoft Windows operating system detects a violation of the software license agreement. A user might have tried to change the product type of an offline system or change the trial period of an evaluation unit of Windows. For more information about the specific violation, see the parameter list.
0x0000009B	UDFS_FILE_SYSTEM	This bug check indicates that a problem occurred in the UDF file system.	The UDFS_FILE_SYSTEM bug check might be caused disk corruption. Corruption in the file system or bad blocks (sectors) on the disk can induce this error. Corrupted SCSI and IDE drivers can also adversely affect the system's ability to read and write to the disk and cause the error. This bug check might also occur if nonpaged pool memory is full. If the nonpaged pool memory is full, this error can stop the system. However, during the indexing process, if the amount of available nonpaged pool memory is very low, another kernel-mode driver that requires nonpaged pool memory can also trigger this error.
0x0000009C	MACHINE_CHECK_EXCEPTION	This bug check indicates that a fatal machine check exception has occurred.	Parameter 1 indicates the type of violation. 0x1 0x2 The system abstraction layer (SAL) returned an error for SAL_GET_STATEINFO while processing MCA. 0x3 SAL returned an error for SAL_CLEAR_STATEINFO while it processed MCA. 0x4 Firmware (FW) reported a fatal MCA. 0x5 There are two possible causes: SAL reported a recoverable MCA, but this recovery is not currently supported. SAL generated an MCA but could not produce an error record. 0xB 0xC SAL returned an error for SAL_GET_STATEINFO while processing an INIT event. 0xD 0xE SAL returned an error for SAL_CLEAR_STATEINFO while it processed an INIT event. Usually this bug check occurs only in the following circumstances. WHEA is not fully initialized. All processors that rendezvous have no errors in their registers.

0x0000009E	USER_MODE_HEALTH_MONITOR	This bug check indicates that one or more critical user-mode components failed to satisfy a health check.	Hardware mechanisms, such as watchdog timers, can detect that basic kernel services are not executing. However, resource starvation issues (including memory leaks, lock contention, and scheduling priority misconfiguration) can block critical user-mode components without blocking deferred procedure calls (DPCs) or draining the non-paged pool. Kernel components can extend watchdog timer functionality to user mode by periodically monitoring critical applications. This bug check indicates that a user-mode health check failed in a way that prevents graceful shutdown. This bug check restores critical services by restarting or enabling application failover to other servers. A user-mode hang can also cause this bug check.
0x0000009F	DRIVER_POWER_STATE_FAILURE	This bug check indicates that the driver is in an inconsistent or invalid power state.	The cause is indicated in the first parameter 0x3 A device object has been blocking an IRP for too long a time. 0x4 The power state transition timed out waiting to synchronize with the PnP subsystem. 0x100 The device objects in the devnode inconsistently used DO_POWER_PAGABLE. 0x101 A parent device object has detected that a child device has not set the DO_POWER_PAGABLE bit. 0x500 The device object completed the IRP for the system power state request, but it did not call PoStartNextPowerIrp.
0x000000A0	INTERNAL_POWER_ERROR	This bug check indicates that the power policy manager experienced a fatal error.	See general suggestions above the table.
0x000000A1	PCI_BUS_DRIVER_INTERNAL	This bug check indicates that the PCI Bus driver detected inconsistency problems in its internal structures and could not continue.	See general suggestions above the table.
0x000000A2	MEMORY_IMAGE_CORRUPT	This bug check indicates that corruption has been detected in the image of an executable file in memory.	A cyclic redundancy check (CRC) check on the memory range has failed. On a system wake operation, various regions of memory might be checked to guard against memory failures.
0x000000A3	ACPI_DRIVER_INTERNAL	This bug check indicates that the ACPI driver detected an internal inconsistency.	An inconsistency in the ACPI driver is so severe that continuing to run would cause serious problems. Possible source of this problem is a BIOS error or ACPI driver problems.
0x000000A4	CNSS_FILE_SYSTEM_FILTER	This bug check indicates that a problem occurred in the CNSS file system filter.	The CNSS_FILE_SYSTEM_FILTER bug check might occur because nonpaged pool memory is full. If the nonpaged pool memory is completely full, this error can stop the system. However, during the indexing process, if the amount of available nonpaged pool memory is very low, another kernel-mode driver that requires nonpaged pool memory can also trigger this error. To resolve a nonpaged pool memory depletion problem: Add new physical memory to the computer. This memory sincrease the quantity of nonpaged pool memory available to the kernel.
0x000000A5	ACPI_BIOS_ERROR	This bug check indicates that the Advanced Configuration and Power Interface (ACPI) BIOS of the computer is not fully compliant with the ACPI specification.	The value of Parameter 1 indicates the error. 0x01 ACPI cannot find the System Control Interrupt (SCI) vector in the resources that are handed to it when ACPI is started. 0x02, the ACPI BIOS could not process the resource list for the PCI root buses. 0x03 ACPI tried to run a control method while creating device extensions to represent the ACPI namespace, but this control method failed. 0x04 ACPI evaluated a _PRW and expected to find an integer as a package element. 0x05 ACPI evaluated a _PRW, and the package that came back failed to contain at least two elements. The ACPI specification requires that two elements always be present in a _PRW. 0x06 ACPI tried to find a named object, but it could not find the object. 0x07 ACPI evaluated a method and expected to receive a buffer in return. However, the method returned some other data type. 0x08 ACPI evaluated a method and expected to receive an integer in return. However, the method returned some other data type. 0x09 ACPI evaluated a method and expected to receive a package in return. However, the method returned some other data type.

			0x0A ACPI evaluated a method and expected to receive a string in return. However, the method returned some other data type. 0x0B ACPI cannot find the object that an _EJD string references. 0x0C ACPI provides faulty or insufficient information for dock support. 0x0D ACPI could not find a required method or object in the namespace This bug check code is used if there is no _HID or _ADR present. 0x0E ACPI could not find a required method or object in the namespace for a power resource (or entity other than a "device"). This bug check code is used if there is no _ON, _OFF, or _STA present for a power resource. 0x0F ACPI could not parse the resource descriptor. 0x10, the ACPI BIOS could not determine the system-to-device-state mapping correctly. 0x11 the system could not enter ACPI mode. 0x14 ACPI could not parse the resource descriptor. The length exceeds MAXULONG. 0x15 ACPI had a fatal error when attempting to load a table. 0x16 ACPI had a fatal error when processing an xSDT. An object was declared as a child of a parent that cannot have children.
0x000000A7	BAD_EXHANDLE	This bug check indicates that the kernel-mode handle table detected an inconsistent handle table entry state.	See general suggestions above the table.
0x000000AB	SESSION_HAS_VALID_POOL_ON_EXIT	This bug check indicates that a session unload occurred while a session driver still held memory.	The SESSION_HAS_VALID_POOL_ON_EXIT bug check occurs because a session driver does not free its pool allocations before a session unload. This bug check indicates a bug in Win32k.sys, Atmfd.dll, Rdpdd.dll, or a video driver.
0x000000AC	HAL_MEMORY_ALLOCATION	This bug check indicates that the hardware abstraction layer (HAL) could not obtain sufficient memory.	The HAL could not obtain non-paged memory pool for a system critical requirement. These critical memory allocations are made early in system initialization, and the HAL_MEMORY_ALLOCATION bug check is not expected. This bug check probably indicates some other critical error such as pool corruption or massive consumption.
0x000000AD	VIDEO_DRIVER_DEBUG_REPORT_REQUEST	This bug check indicates that the video port created a non-fatal minidump on behalf of the video driver during run time.	The video port created a non-fatal minidump on behalf of the video driver during run time because the video driver requested a debug report. The VIDEO_DRIVER_DEBUG_REPORT_REQUEST bug check can be caused only by minidump creation, not by the creation of a full dump or kernel dump.
0x000000B4	VIDEO_DRIVER_INIT_FAILURE	This indicates that Windows was unable to enter graphics mode.	The system was not able to go into graphics mode because no display drivers were able to start. This usually occurs when no video miniport drivers are able to load successfully.
0x000000B8	ATTEMPTED_SWITCH_FROM_DPC	This indicates that an illegal operation was attempted by a delayed procedure call (DPC) routine.	A wait operation, attach process, or yield was attempted from a DPC routine. This is an illegal operation.
0x000000B9	CHIPSET_DETECTED_ERROR	The chipset has detected an error.	See general suggestions above the table.
0x000000BA	SESSION_HAS_VALID_VIEWS_ON_EXIT	This indicates that a session driver still had mapped views when the session unloaded.	This error is caused by a session driver not unmapping its mapped views prior to a session unload. This indicates a bug in win32k.sys, atmfd.dll, rdpdd.dll, or a video driver.
0x000000BB	NETWORK_BOOT_INITIALIZATION_FAILED	This indicates that Windows failed to successfully boot off a network.	This error is caused when Windows is booting off a network, and a critical function fails during I/O initialization.
0x000000BC	NETWORK_BOOT_DUPLICATE_ADDRESS	This indicates that a duplicate IP address was assigned to this machine while booting off a network.	This error indicates that when TCP/IP sent out an ARP for its IP address, it got a response from another machine indicating a duplicate IP address. When Windows is booting off a network, this is a fatal error.
0x000000BE	ATTEMPTED_WRITE_TO_READONLY_MEMORY	This is issued if a driver attempts to write to a read-only memory segment.	If the driver responsible for the error can be identified, its name is printed on the blue screen and stored in memory at the location (PUNICODE_STRING) KiBugCheckDriver.
0x000000BF	MUTEX_ALREADY_OWNED	This indicates that a thread attempted to acquire ownership of a mutex it already owned.	See general suggestions above the table.
0x000000C1	SPECIAL_POOL_DETECTED_MEMORY_CORRUPTION	This indicates that the driver wrote to an invalid section of the special pool.	A driver has written to an invalid section of the special pool. Obtain a backtrace of the current thread. This backtrace will usually reveal the source of the error.

0x000000C2	BAD_POOL_CALLER	This indicates that the current thread is making a bad pool request.	An invalid pool request has been made by the current thread. Parameter 1 indicates the type of violation 0x00 The current thread requested a zero-byte pool allocation. 0x01 0x02 0x04 The pool header has been corrupted. 0x06 The current thread attempted to free the pool, which was already freed. 0x07 The current thread attempted to free the pool, which was already freed. 0x08 The current thread attempted to allocate the pool at an invalid IRQL. 0x09 The current thread attempted to free the pool at an invalid IRQL. 0x0A The current thread attempted to free pool memory by using the wrong tag. (The memory might belong to another component.) 0x0B 0x0C 0x0D The current thread attempted to release a quota on a corrupted pool allocation. 0x40 The current thread attempted to free the kernel pool at a user-mode address. 0x41 The current thread attempted to free a non-allocated nonpaged pool address. 0x42 0x43 The current thread attempted to free a virtual address that was never in any pool. 0x44 The current thread attempted to free a non-allocated nonpaged pool address. 0x46 The current thread attempted to free an invalid pool address. 0x47 The current thread attempted to free a non-allocated nonpaged pool address. 0x48 The current thread attempted to free a non-allocated paged pool address. 0x50 The current thread attempted to free a non-allocated paged pool address. 0x60 The current thread attempted to free an invalid contiguous memory address. (The caller of MmFreeContiguousMemory is passing a bad pointer.) 0x99 The current thread attempted to free pool with an invalid address. (This code can also indicate corruption in the pool header.) 0x9A The current thread marked an allocation request MUST_SUCCEED. (This pool type is no longer supported.) 0x9B The current thread attempted to allocate a pool with a tag of 0 (This would be untrackable, and possibly corrupt the existing tag tables.) 0x9C The current thread attempted to allocate a pool with a tag of "BIG". 0x9D The current thread attempted to allocate a pool with a tag that does not contain any letters or digits. Using such tags makes tracking pool issues difficult. 0x41286 The current thread attempted to free a paged pool address in the middle of an allocation.
0x000000C4	DRIVER_VERIFIER_DETECTED_VIOLATION	This is the general bug check code for fatal errors found by Driver Verifier.	Driver Verifier detects driver errors at run time. Thee deep check of errors may prevent startup and others operation in this case you can disable drive verifier (see drive verifier section).
0x000000C5	DRIVER_CORRUPTED_EXPOOL	This indicates that the system attempted to access invalid memory at a process IRQL that was too high.	The kernel attempted to access pageable memory (or perhaps completely invalid memory) when the IRQL was too high. The ultimate cause of this problem is almost certainly a driver that has corrupted the system pool. In most cases, this bug check results if a driver corrupts a small allocation (less than PAGE_SIZE). Larger allocations result in bug check 0xD0 (DRIVER_CORRUPTED_MMPOOL).
0x000000C6	DRIVER_CAUGHT_MODIFYING_FREED_POOL	This indicates that the driver attempted to access a freed memory pool.	The faulty component will be displayed in the current kernel stack. This driver should be either replaced or debugged.
0x000000C7	TIMER_OR_DPC_INVALID	This is issued if a kernel timer or delayed procedure call (DPC) is found somewhere in memory where it is not permitted.	This condition is usually caused by a driver failing to cancel a timer or DPC before freeing the memory where it resides.
0x000000C8	IRQL_UNEXPECTED_VALUE	This indicates that the processor's IRQL is not what it should be at this time.	This error is usually caused by a device driver or another lower-level program that changed the IRQL for some period and did not restore the original IRQL at the end of

			that period. For example, the routine may have acquired a spin lock and failed to release it.
0x000000C9	DRIVER_VERIFIER_IOMANAGER_VIOLATION	This is the bug check code for all Driver Verifier I/O Verification violations.	Parameter 1 identifies the type of violation. 0x01 The driver attempted to free an object whose type is not IO_TYPE_IRP. 0x02 The driver attempted to free an IRP that is still associated with a thread. 0x03 The driver passed IoCallDriver an IRP Type not equal to IRP_TYPE. 0x04 The driver passed IoCallDriver an invalid device object. 0x05 The IRQL changed during a call to the driver dispatch routine. 0x06 The driver called IoCompleteRequest with a status marked as pending (or equal to -1). 0x07 The driver called IoCompleteRequest while its cancel routine was still set. 0x08 The driver passed IoBuildAsynchronousFsdRequest an invalid buffer. 0x09 The driver passed IoBuildDeviceIoControlRequest an invalid buffer. 0x0A The driver passed IoInitializeTimer a device object with an already-initialized timer. 0x0C The driver passed an I/O status block to an IRP, but this block is allocated on a stack which has already unwound past that point. 0x0D The driver passed a user event to an IRP, but this event is allocated on a stack which has already unwound past that point. 0x0E The driver called IoCompleteRequest with IRQL > DISPATCH_LEVEL. 0x0F The driver sent a create request with a file object that has been closed, or that had its open canceled.
0x000000CA	PNP_DETECTED_FATAL_ERROR	This indicates that the Plug and Play Manager encountered a severe error, probably as a result of a problematic Plug and Play driver.	Parameter 1 identifies the type of violation. 0x1 Duplicate PDO: A specific instance of a driver has enumerated multiple PDOs with identical device ID and unique IDs. 0x2 Invalid PDO: An API which requires a PDO has been called with random memory, or with an FDO, or with a PDO which hasn't been initialized. (An uninitialized PDO is one that has not been returned to Plug and Play by QueryDeviceRelation or QueryBusRelations.) 0x3 Invalid ID: An enumerator has returned an ID which contains illegal characters or isn't properly terminated. (IDs must contain only characters in the ranges 0x20 - 0x2B and 0x2D - 0x7F.) 0x4 Invalid enumeration of deleted PDO: An enumerator has returned a PDO which it had previously deleted using IoDeleteDevice. 0x5 PDO freed while linked in devnode tree: The object manager reference count on a PDO dropped to zero while the devnode was still linked in the tree. (This usually indicates that the driver is not adding a reference when returning the PDO in a query IRP.) 0x8 NULL pointer returned as a bus relation: One or more of the devices present on the bus is a NULL PDO. 0x9 Invalid connection type passed to IoDisconnectInterruptEx: A driver has passed an invalid connection type to IoDisconnectInterruptEx. The connection type passed to this routine must match the one returned by a corresponding successful call to IoConnectInterruptEx. 0xA Incorrect notify callback behavior: A driver failed to preserve IRQL or combined APC disable count across a Plug 'n' Play notification. 0xB Deleted PDO reported as relation: One of the removal relations for the device being removed has already been deleted.
0x000000CB	DRIVER_LEFT_LOCKED_PAGES_IN_PROCESS	This indicates that a driver or the I/O manager failed to release locked pages after an I/O operation.	This bug check is issued only if the registry value \\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\TrackLockedPages is equal to DWORD 1. If this value is not set, the system will issue the less-informative bug check 0x76 (PROCESS_HAS_LOCKED_PAGES). This bug check can also be issued by Driver Verifier when the Pool Tracking option is enabled.

0x000000CC	PAGE_FAULT_IN_FREED_SPECIAL_POOL	This indicates that the system has referenced memory which was earlier freed.	The Driver Verifier Special Pool option has caught the system accessing memory which was earlier freed. This usually indicates a system-driver synchronization problem. For information about the special pool, consult the Driver Verifier section of the Windows Driver Kit.
0x000000CD	PAGE_FAULT_BEYOND_END_OF_ALLOCATION	This indicates that the system accessed memory beyond the end of some driver's pool allocation.	The driver allocated n bytes of memory from the special pool. Subsequently, the system referenced more than n bytes from this pool. This usually indicates a system-driver synchronization problem. For information about the special pool, consult the Driver Verifier section of the Windows Driver Kit.
0x000000CE	DRIVER_UNLOADED_WITHOUT_CANCELLING_PENDING_OPERATIONS	This indicates that a driver failed to cancel pending operations before unloading.	This driver failed to cancel lookaside lists, DPCs, worker threads, or other such items before unload.
0x000000CF	TERMINAL_SERVER_DRIVER_MADE_INCORRECT_MEMORY_REFERENCE	This indicates that a driver has been incorrectly ported to the terminal server.	The driver is referencing session space addresses from the system process context. This probably results from the driver queuing an item to a system worker thread. This driver needs to comply with Terminal Server's memory management rules.
0x000000D0	DRIVER_CORRUPTED_MMPOOL	This indicates that the system attempted to access invalid memory at a process IRQL that was too high.	The kernel attempted to access pageable memory (or perhaps completely invalid memory) when the IRQL was too high. The ultimate cause of this problem is almost certainly a driver that has corrupted the system pool. In most cases, this bug check results if a driver corrupts a large allocation (PAGE_SIZE or larger). Smaller allocations result in bug check 0xC5 (DRIVER_CORRUPTED_EXPOOL).
0x000000D1	DRIVER_IRQL_NOT_LESS_OR_EQUAL	This indicates that a kernel-mode driver attempted to access pageable memory at a process IRQL that was too high.	A driver tried to access an address that is pageable (or that is completely invalid) while the IRQL was too high. This bug check is usually caused by drivers that have used improper addresses. If the first parameter has the same value as the fourth parameter, and the third parameter indicates an execute operation, this bug check was likely caused by a driver that was trying to execute code when the code itself was paged out. Possible causes for the page fault include the following: The function was marked as pageable and was running at an elevated IRQL (which includes obtaining a lock). The function call was made to a function in another driver, and that driver was unloaded. The function was called by using a function pointer that was an invalid pointer.
0x000000D2	BUGCODE_ID_DRIVER	This indicates that a problem occurred with an NDIS driver.	Before this bug check occurs, a message is sent to the DbgPrint buffer. If a debugger is connected, this message will be displayed. This message indicates the type of violation. The meanings of the bug check parameters depend on this message.
0x000000D3	DRIVER_PORTION_MUST_BE_NONPAGED	This indicates that the system attempted to access pageable memory at a process IRQL that was too high.	This bug check is usually caused by drivers that have incorrectly marked their own code or data as pageable.
0x000000D4	SYSTEM_SCAN_AT_RAISED_IRQL_CAUGHT_IMPROPER_DRIVER_UNLOAD	This indicates that a driver did not cancel pending operations before unloading.	This driver failed to cancel lookaside lists, DPCs, worker threads, or other such items before unload. Subsequently, the system attempted to access the driver's former location at a raised IRQL.
0x000000D5	DRIVER_PAGE_FAULT_IN_FREED_SPECIAL_POOL	This indicates that a driver has referenced memory which was earlier freed.	The Driver Verifier Special Pool option has caught the driver accessing memory which was earlier freed. For information about the special pool, consult the Driver Verifier section of the Windows Driver Kit.
0x000000D6	DRIVER_PAGE_FAULT_BEYOND_END_OF_ALLOCATION	This indicates the driver accessed memory beyond the end of its pool allocation.	The driver allocated n bytes of memory and then referenced more than n bytes. The Driver Verifier Special Pool option detected this violation. This cannot be protected by a try except handler it can only be protected by a probe.
0x000000D7	DRIVER_UNMAPPING_INVALID_VIEW	This indicates a driver is trying to unmap an address that was not mapped.	The driver that caused the error can be determined from the stack trace.
0x000000D8	DRIVER_USED_EXCESSIVE_PTES	This indicates that there are no more system page table entries (PTE) remaining.	This is usually caused by a driver not cleaning up its memory use properly. Parameter 1 shows the driver which has consumed the most PTEs. The call stack will reveal which driver actually caused the bug check.
0x000000D9	LOCKED_PAGES_TRACKER_CORRUPTION	This indicates that the internal locked-page tracking structures have been corrupted.	The error is indicated by the value of Parameter 1. 0x01 The MDL is being inserted twice on the same process list. 0x02 The MDL is being inserted twice on the systemwide list. 0x03 The MDL was found twice in the process list when

			being freed. 0x04 The MDL was found in the systemwide list on free after it was removed.
0x000000DA	SYSTEM_PTE_MISUSE	This indicates that a page table entry (PTE) routine has been used in an improper way.	The error is indicated by the value of Parameter 1. 0x01 The mapping being freed is a duplicate. 0x02 The number of mappings being freed is incorrect. 0x03 The mapping address being freed is incorrect. 0x04 The first page of the mapped MDL has changed since the MDL was mapped. 0x05 The start virtual address in the MDL being freed has changed since the MDL was mapped. 0x06 The MDL being freed was never (or is currently not) mapped. 0x07 The mapping range is being double-allocated. 0x08 The caller is asking to free an incorrect number of mappings. 0x09 The caller is asking to free several mappings, but at least one of them is not allocated. 0x0A The caller is asking to allocate zero mappings. 0x0B The mapping list was already corrupt at the time of this allocation. The corrupt mapping is located below the lowest possible mapping address. 0x0C The mapping list was already corrupt at the time of this allocation. The corrupt mapping is located above the lowest possible mapping address. 0x0D The caller is trying to free zero mappings. 0x0E The caller is trying to free mappings, but the guard mapping has been overwritten. 0x0F The caller is trying to free a non-existent mapping. The non-existent mapping is located below the lowest possible mapping address. 0x10 The caller is trying to free a non-existent mapping. The non-existent mapping is located above the highest possible mapping address. 0x11 The caller is trying to free a non-existent mapping. The non-existent mapping is at the base of the mapping address space. 0x100 The caller requested 0 mappings. 0x101 A caller is trying to free a mapping address range that it does not own. 0x102 The mapping address space that the caller is trying to free is apparently empty. 0x103 The mapping address space that the caller is trying to free is still reserved. MmUnmapReservedMapping must be called before MmFreeMappingAddress. 0x104 The caller is attempting to map an MDL to a mapping address space that it does not own. 0x105 The caller is attempting to map an MDL to an invalid mapping address space. The caller has mostly likely specified an invalid address. 0x107 The caller is attempting to map an MDL to a mapping address space that has not been properly reserved. The caller should have called 0x108 The caller is attempting to unmap a locked mapping address space that it does not own. 0x109 The caller is attempting to unmap a locked virtual address space that is apparently empty. 0x10A The caller is attempting to unmap more mappings than actually exist in the locked mapping address space. 0x10B The caller is attempting to unmap a portion of a locked virtual address space that is not currently mapped 0x10C The caller is not unmapping the entirety of the locked mapping address space. 0x200 The caller is attempting to reserve a mapping address space that contains no mappings. 0x201 0x202 One of the mappings that the caller is attempting to reserve has already been reserved. 0x300 The caller is attempting to release a mapping address space that contains no mappings. 0x301 The caller is attempting to release a mapping that it is not permitted to release. 0x302 The caller is attempting to release a system address that is not currently mapped. 0x303 The caller is attempting to release a mapping address range that was not reserved. 0x304 The caller is attempting to release a mapping

			address range that begins in the middle of a different allocation. 0x305 The caller is attempting to release the wrong number of mappings. 0x306 One of the mappings that the caller is attempting to release is already free. 0x400 The caller is trying to free an I/O space mapping that the system is unaware of.
0x000000DB	DRIVER_CORRUPTED_SYSPTES	This indicates that an attempt was made to touch memory at an invalid IRQL, probably due to corruption of system PTEs.	A driver tried to access pageable (or completely invalid) memory at too high of an IRQL. This bug check is almost always caused by drivers that have corrupted system PTEs. If this bug check occurs, the culprit can be detected by editing the registry. In the \\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management registry key, create or edit the TrackPtes value, and set it equal to DWORD 3. Then reboot. The system will then save stack traces, and if the driver commits the same error, the system will issue bug check 0xDA (SYSTEM_PTE_MISUSE). Then the stack trace will identify the driver that caused the error.
0x000000DC	DRIVER_INVALID_STACK_ACCESS	This indicates that a driver accessed a stack address that lies below the stack pointer of the stack's thread.	See general suggestions above the table.
0x000000DE	POOL_CORRUPTION_IN_FILE_AREA	This indicates that a driver has corrupted pool memory that is used for holding pages destined for disk.	When the Memory Manager dereferenced the file, it discovered this corruption in pool memory.
0x000000DF	IMPERSONATING_WORKER_THREAD	This indicates that a workitem did not disable impersonation before it completed.	A worker thread was impersonating another process, and failed to disable impersonation before it returned.
0x000000E0	ACPI_BIOS_FATAL_ERROR	This indicates that one of your computer components is faulty.	Your computer's BIOS has reported that a component in the system is so faulty that there is no way for Windows to operate. The BIOS is indicating that there is no alternative but to issue a bug check.
0x000000E1	WORKER_THREAD_RETURNED_AT_BAD_IRQL	This indicates that a worker thread completed and returned with IRQL >= DISPATCH_LEVEL.	See general suggestions above the table.
0x000000E2	MANUALLY_INITIATED_CRASH	This indicates that the user deliberately initiated a crash dump from either the kernel debugger or the keyboard.	See general suggestions above the table.
0x000000E3	RESOURCE_NOT_OWNED	This indicates that a thread tried to release a resource it did not own.	See general suggestions above the table.
0x000000E4	WORKER_INVALID	This indicates that memory that should not contain an executive work item does contain such an item, or that a currently active work item was queued.	See general suggestions above the table.
0x000000E6	DRIVER_VERIFIER_DMA_VIOLATION	This is the bug check code for all Driver Verifier DMA Verification violations.	Parameter 1 identifies the exact violation. 0x00 This code can represent two kinds of errors: -The driver tried to flush too many bytes to the end of the map register file. The number of bytes permitted and the number of bytes attempted are displayed. -Windows has run out of contiguous map registers. The number of map registers needed and the largest block of contiguous map registers is displayed. 0x01 The performance counter has decreased. The old and new values of the counter are displayed. 0x02 The performance counter has increased too fast. The counter value is displayed in the debugger. 0x03 The driver freed too many DMA common buffers. Usually this means it freed the same buffer two times. 0x04 The driver freed too many DMA adapter channels. Usually this means it freed the same adapter channel two times. 0x05 The driver freed too many DMA map registers. Usually this means it freed the same map register two times. The number of active map registers is displayed. 0x06 The driver freed too many DMA scatter/gather lists. Usually this means it freed the same scatter/gather list two times. The number of lists allocated and the number of lists freed is displayed. 0x07 The driver tried to release the adapter without first freeing all its common buffers. The adapter address and the number of remaining buffers is displayed. 0x08 The driver tried to release the adapter without first freeing all adapter channels, common buffers, or scatter/gather lists. The adapter address and the number of remaining items is displayed.

			0x09 The driver tried to release the adapter without first freeing all map registers. The adapter address and the number of remaining map registers is displayed. 0x0A The driver tried to release the adapter without first freeing all its scatter/gather lists. The adapter address and the number of remaining scatter/gather lists is displayed. 0x0B HV_TOO_MANY_ADAPTER_CHANNELSThe driver has allocated too many adapter channels at the same time. . (Only one adapter channel is permitted per adapter.) 0x0C The driver tried to allocate too many map registers at the same time. The number requested and the number allowed are displayed. 0x0D The driver did not flush its adapter buffers. The number of bytes that the driver tried to map and the maximum number of bytes allowed are displayed. 0x0E The driver tried a DMA transfer without locking the buffer. The buffer in question was in paged memory. The address of the MDL is displayed. 0x0F The driver or the hardware wrote outside its allocated DMA buffer. The nature of the error (overrun or underrun) is displayed, as well as the relevant addresses. 0x10 The driver tried to free its map registers while some were still mapped. The number of map registers still mapped is displayed. 0x11 The driver has too many outstanding reference counts for the adapter. The number of reference counts and the adapter address are displayed. 0x13 The driver called a DMA routine at an improper IRQL. The required IRQL and the actual IRQL are displayed. 0x14 The driver called a DMA routine at an improper IRQL. The required IRQL and the actual IRQL are displayed. 0x15 The driver tried to allocate too many map registers. The number requested and the number allowed are displayed. 0x16 The driver tried to flush a buffer that is not mapped. The address of the buffer is displayed. 0x18 The driver tried a DMA operation by using an adapter that was already released and no longer exists. The adapter address is displayed. 0x19 The driver passed a null DMA_ADAPTER value to a HAL routine. 0x1B The driver passed an address and MDL to a HAL routine. However, this address is not within the bounds of this MDL. The address passed and the address of the MDL are displayed. 0x1D The driver tried to map an address range that was already mapped. The address range and the current mapping for that range are displayed. 0x1E The driver called HalGetAdapter. This function is obsolete -- you must use IoGetDmaAdapter instead. 0x1F HV_BAD_MDLThe driver referenced an invalid system address -- either before the first MDL, or after the end of the first MDL, or by using a transfer length that is longer than the MDL buffer and crosses a page boundary within the MDL. . Either the invalid address and the first MDL address, or the MDL address and the extra transfer length are displayed. 0x20 The driver tried to flush a map register that hasn't been mapped. The map register base, flushing address, and MDL are displayed. 0x21 The driver tried to map a zero-length buffer for transfer.
0x000000E7	INVALID_FLOATING_POINT_STATE	This indicates that a thread's saved floating-point state is invalid.	Parameter 1 indicates which validity check failed. 0x0 The saved context flags field is invalid. Either FLOAT_SAVE_VALID is not set, or some reserved bits are nonzero. 0x1 The current processor's IRQL is not the same as when the floating-point context was saved. 0x2 The saved context does not belong to the current thread.

0x000000E8	INVALID_CANCEL_OF_FILE_OPEN	This indicates that an invalid file object was passed to IoCancelFileOpen.	The file object passed to IoCancelFileOpen is invalid. It should have reference of one. The driver that called IoCancelFileOpen is at fault.
0x000000E9	ACTIVE_EX_WORKER_THREAD_TERMINATION	This indicates that an active executive worker thread is being terminated.	An executive worker thread is being terminated without having gone through the worker thread rundown code. This is forbidden; work items queued to the ExWorkerQueue must not terminate their threads. A stack trace should indicate the cause.
0x000000EA	THREAD_STUCK_IN_DEVICE_DRIVER	This indicates that a thread in a device driver is endlessly spinning.	A device driver is spinning in an infinite loop, most likely waiting for hardware to become idle. This usually indicates problem with the hardware itself, or with the device driver programming the hardware incorrectly. Frequently, this is the result of a bad video card or a bad display driver. Use the .thread (Set Register Context) command together with Parameter 1. Then use kb (Display Stack Backtrace) to find the location where the thread is stuck. If the kernel debugger is already connected and running when Windows detects a time-out condition. Then DbgBreakPoint will be called instead of KeBugCheckEx. A detailed message will be printed to the debugger. See Sending Output to the Debuggefor more information. This message will include what would have been the bug check parameters. Because no actual bug check was issued, the .bugcheck (Display Bug Check Data) command will not be useful. The four parameters can also be retrieved from Watchdog's global variables by using dd watchdog!g_WdBugCheckData L5" on a 32-bit system, or dq watchdog!g_WdBugCheckData L5" on a 64-bit system. Debugging this error in an interactive manner such as this will enable you to find an offending thread, set breakpoints in it, and then use g (Go) to return to the spinning code to debug it further. On multiprocessor machines (OS build 3790 or earlier), you can hit a time out if the spinning thread is interrupted by a hardware interrupt and an ISR or DPC routine is running at the time of the bug check. This is because the time out's work item can be delivered and handled on the second CPU and the same time. If this occurs, you must look deeper at the offending thread's stack to determine the spinning code which caused the time out to occur. Use the dds (Display Words and Symbols) command to do this.
0x000000EB	DIRTY_MAPPED_PAGES_CONGESTION	This indicates that no free pages are available to continue operations.	The file system driver stack has deadlocked and most of the modified pages are destined for the file system. Because the file system is non-operational, the system has crashed because none of the modified pages can be reused without losing data. Any file system or filter driver in the stack may be at fault. To see general memory statistics, use the !vm 3 extension. This bug check can occur for any of the following reasons: A driver has blocked, deadlocking the modified or mapped page writers. Examples of this include mutex deadlocks or accesses to paged out memory in file system drivers or filter drivers. This indicates a driver bug. If Parameter 1 or Parameter 2 is large, this is a possibility. Use !vm 3. A storage driver is not processing requests. Examples of this are stranded queues and unresponsive drives. This indicates a driver bug. If Parameter 1 or Parameter 2 is large, this is a possibility. Use !process 0 7. Windows Server only: Not enough pool is available for the storage stack to write out modified pages. This indicates a driver bug. If Parameter 3 is small, this is a possibility. Use !vm and !poolused 2.
0x000000EC	SESSION_HAS_VALID_SPECIAL_POOL_ON_EXIT	This indicates that a session unload occurred while a session driver still held memory.	This error is caused by a session driver not freeing its special pool allocations prior to a session unload. This indicates a bug in win32k.sys, atmfd.dll, rdpdd.dll, or a video driver.

0x000000ED	UNMOUNTABLE_BOOT_VOLUME	This indicates that the I/O subsystem attempted to mount the boot volume and it failed.	See general suggestions above the table.
0x000000EF	CRITICAL_PROCESS_DIED	This indicates that a critical system process died.	See general suggestions above the table.
0x000000F1	SCSI_VERIFIER_DETECTED_VIOLATION	This is the bug check code for all Driver Verifier SCSI Verification violations.	Parameter 1 identifies the type of violation. 0x1000 The miniport driver passed bad arguments to ScsiPortInitialize. 0x1001 The miniport driver called ScsiPortStallExecution and specified a delay greater than 0.1 second, stalling the processor too long. 0x1002 A miniport routine called by the port driver took longer than 0.5 second to execute. (0.5 seconds is the limit for most routines. However, the HwInitialize routine is allowed 5 seconds, and the FindAdapter routine is exempt.) 0x1003 The miniport driver completed a request more than once. 0x1004 The miniport driver completed a request with an invalid SRB status. 0x1005 The miniport driver called ScsiPortNotification to ask for NextLuRequest, but an untagged request is still active. 0x1006 The miniport driver passed an invalid virtual address to ScsiPortGetPhysicalAddress. (This usually means the address supplied doesn't map to the common buffer area.) 0x1007 The reset hold period for the bus ended, but the miniport driver still has outstanding requests. 0x2001 The Storport miniport driver called StorPortStallExecution and specified a delay longer than 0.1 second, stalling the processor for an excessive length of time. 0x2002 StorPortGetUncachedExtension was not called from the miniport driver's HwStorFindAdapter routine. The StorPortGetUncachedExtension routine can only be called from the miniport driver's HwStorFindAdapter routine and only for a bus-master adapter. A Storport miniport driver must set the SrbExtensionSize of the HW_INITIALIZATION_DATA (Storport) structure before calling StorPortGetUncachedExtension. 0x2003 An invalid address was passed to the StorPortGetDeviceBase routine. The StorPortGetDeviceBase routine supports only those addresses that were assigned to the driver by the system Plug and Play (PnP) manager. 0x2004 The Storport miniport driver completed the same I/O request more than once. 0x2005 The Storport miniport driver passed an invalid virtual address to one of the StorPortReadxxx or StorPortWritexxx routines. This usually means the address supplied doesn't map to the common buffer area. The specified Register or Port must be in mapped memory-space range returned by StorPortGetDeviceBase routine.
0x000000F3	DISORDERLY_SHUTDOWN	This indicates that Windows was unable to shut down correctly due to lack of memory.	Windows attempted to shut down, but there were no free pages available to continue operations. Because applications were not terminated and drivers were not unloaded, they continued to access pages even after the modified writer had terminated. This causes the system to run out of pages, since the page files could be used.
0x000000F4	CRITICAL_OBJECT_TERMINATION	This indicates that a process or thread crucial to system operation has unexpectedly exited or been terminated.	Several processes and threads are necessary for the operation of the system. When they are terminated for any reason, the system can no longer function.
0x000000F5	FLTMGR_FILE_SYSTEM	This indicates that an unrecoverable failure occurred in the Filter Manager.	The cause of the problem is indicated by the value of Parameter 1. 0x66 The minifilter returned FLT_PREOP_SUCCESS_WITH_CALLBACK or FLT_PREOP_SYNCHRONIZE from a preoperation callback, but did not register a corresponding postoperation callback. 0x67 An internal object ran out of space, and the system is unable to allocate new space. 0x68 A FLT_FILE_NAME_INFORMATION structure was dereferenced too many times.

			0x6A The file-open or file-create request could not be canceled, because one or more handles have been created for the file. 0x6B Invalid BACKPOCKET IRPCTRL state. 0x6C Too many nested PageFaults for BACKPOCKETED IRPCTR. 0x6D The context structure was dereferenced too many times. This means that the reference count on the Filter Manager's CONTEXT_NODE structure went to zero while it was still attached to its associated object. 0x6E The context structure was referenced after being freed.
0x000000F6	PCI_VERIFIER_DETECTED_VIOLATION	This indicates that an error occurred in the BIOS or another device being verified by the PCI driver.	The PCI driver detected an error in a device or BIOS being verified.
0x000000F7	DRIVER_OVERRAN_STACK_BUFFER	This indicates that a driver has overrun a stack-based buffer.	A driver overran a stack-based buffer (or local variable) in a way that would have overwritten the function's return address and jumped back to an arbitrary address when the function returned. This is the classic "buffer overrun" hacking attack. The system has been brought down to prevent a malicious user from gaining complete control of it.
0x000000F8	RAMDISK_BOOT_INITIALIZATION_FAILED	This indicates that an initialization failure occurred while attempting to boot from the RAM disk.	See general suggestions above the table.
0x000000F9	DRIVER_RETURNED_STATUS_REPARSE_FOR_VOLUME_OPEN	This indicates that a driver returned STATUS_REPARSE to an IRP_MJ_CREATE request with no trailing names.	STATUS_REPARSE should be returned only for IRP_MJ_CREATE requests with trailing names, as that indicates the driver is supporting name spaces.
0x000000FA	HTTP_DRIVER_CORRUPTED	This indicates that the HTTP kernel driver (Http.sys) has reached a corrupted state and cannot recover.	A work item is invalid. This will eventually result in thread pool corruption and an access violation.
0x000000FC	ATTEMPTED_EXECUTE_OF_NOEXECUTE_MEMORY	This indicates that an attempt was made to execute non-executable memory.	When possible, the Unicode string of the driver name that attempted to execute non-executable memory is printed on the bug check screen and is also saved in KiBugCheckDriver. Otherwise, the driver in question can often be found by running a stack trace and then reviewing the current instruction pointer.
0x000000FD	DIRTY_NOWRITE_PAGES_CONGESTION	This indicates that there are no free pages available to continue basic system operations.	This bug check usually occurs because the component that owns the modified non-writeable pages failed to write out these pages after marking the relevant files as "do not write" to memory management. This indicates a driver bug.
0x000000FE	BUGCODE_USB_DRIVER	This indicates that an error has occurred in a universal serial bus (USB) driver.	0x1 Parameter 1 identifies the type of violation. An internal error has occurred in the USB stack. 0x2 The USB client driver has submitted a URB that is still attached to another IRP pending in the bus driver. 0x3 The USB miniport driver has generated a bug check. This usually happens in response to a hardware failure. 0x4 The caller has submitted an IRP that is already pending in the USB bus driver. 0x5 A hardware failure has occurred because of a bad physical address found in a hardware data structure. 0x6 An internal data structure (object) is corrupted. 0x7 Please see the provided message string for detailed information. 0x8 An IRP was received by the hub driver that it does not expect or has not registered for, a fatal PDO trap happened or the device hasn’t responded in the timeout time.
0x000000FF	RESERVE_QUEUE_OVERFLOW	This indicates that an attempt was made to insert a new item into a reserve queue, causing the queue to overflow.	See general suggestions above the table.
0x00000100	LOADER_BLOCK_MISMATCH	This indicates that either the loader block is invalid, or it does not match the system that is being loaded.	See general suggestions above the table.
0x00000101	CLOCK_WATCHDOG_TIMEOUT	This indicates that an expected clock interrupt on a secondary processor, in a multi-processor system, was not received within the allocated interval.	The specified processor is not processing interrupts. Typically, this occurs when the processor is nonresponsive or is deadlocked.
0x00000103	MUP_FILE_SYSTEM	This bug check indicates that the multiple UNC provider (MUP) has encountered invalid or unexpected data. As a result, the MUP cannot channel a remote file system request to a network redirector, the Universal Naming Convention (UNC) provider.	The MUP maintains context information on a per-file object basis for all file objects it handles. Parameter 1 identifies the type of violation. 0x1 The MUP could not locate the file context that corresponds to a file object. This typically indicates that the MUP is seeing an I/O request for a file object for which MUP did not see a corresponding IRP_MJ_CREATE

			request. The likely cause of this bug check is a filter driver error. 0x2 A file context is known to exist for the file object, but was not what was expected (for example, it might be NULL). 0x3 The IRP completion status was unexpected or invalid. This bug check occurs only when you are using a Checked Build of Windows and should only be caused by file system filter drivers that are attached to legacy network redirectors. Legacy redirectors use FsRtlRegisterUncProvider to register with MUP. This bug check detects filter drivers that return an NTSTATUS that is not STATUS_SUCCESS in IRP_MJ_CLEANUP or IRP_MJ_CLOSE requests. 0x4 An I/O operation was started on a file object before the create request for the file object was completed.
0x00000104	AGP_INVALID_ACCESS	This indicates that the GPU wrote to a range of Accelerated Graphics Port (AGP) memory that had not previously been committed.	Typically, this bug check is caused by an unsigned or improperly tested video driver. It can also be caused by an old BIOS.
0x00000105	AGP_GART_CORRUPTION	This indicates that the Graphics Aperture Remapping Table (GART) is corrupt.	This bug check is typically caused by improper direct memory access (DMA) by a driver.
0x00000106	AGP_ILLEGALLY_REPROGRAMMED	This indicates that the Accelerated Graphics Port (AGP) hardware has been reprogrammed by an unauthorized agent.	This bug check is typically caused by an unsigned, or improperly tested, video driver. Check the video manufacturer's Web site for updated display drivers or use VGA mode.
0x00000108	THIRD_PARTY_FILE_SYSTEM_FAILURE	This indicates that an unrecoverable problem has occurred in a third-party file system or file system filter.	One possible cause of this bug check is disk corruption. Corruption in the third-party file system or bad blocks (sectors) on the hard disk can induce this error. Corrupted SCSI and IDE drivers can also adversely affect the Windows operating system's ability to read and write to disk, thus causing the error. Another possible cause is depletion of nonpaged pool memory. If the nonpaged pool is completely depleted, this error can stop the system.
0x00000109	CRITICAL_STRUCTURE_CORRUPTION	This indicates that the kernel has detected critical kernel code or data corruption.	There are generally three different causes for this bug check: -A driver has inadvertently, or deliberately, modified critical kernel code or data. Microsoft Windows Server 2003 with Service Pack 1 (SP1) and later versions of Windows for x64-based computers do not allow the kernel to be patched except through authorized Microsoft-originated hot patches. For more information, see Patching Policy for x64-based Systems. -A developer attempted to set a normal kernel breakpoint using a kernel debugger that was not attached when the system was started. Normal breakpoints (bp) can only be set if the debugger is attached at start time. Processor breakpoints (ba) can be set at any time. -A hardware corruption occurred. For example, the kernel code or data could have been stored in memory that failed.
0x0000010A	APP_TAGGING_INITIALIZATION_FAILED	The system was not able to initialize the app tagging service.	See general suggestions above the table.
0x0000010C	FSRTL_EXTRA_CREATE_PARAMETER_VIOLATION	This indicates that a violation was detected in the File system Run-time library (FsRtl) Extra Create Parameter (ECP) package.	The value of Parameter 1 indicates the type of violation. 0x1 The ECP signature is invalid, due to either a bad pointer or memory corruption. 0x2 The ECP has undefined flags set. 0x3 The ECP was not allocated by the FsRtl. 0x4 The ECP has flags set that are illegal for a parameter passed by a create caller. 0x5 The ECP is corrupted; its size is smaller than the header size. 0x6 The ECP that is being freed has non-empty list pointers; it might still be part of an ECP list. 0x11 The ECP list signature is invalid, due to either a bad pointer or memory corruption. 0x12 The ECP list has undefined flags set. 0x13 The ECP list was not allocated by the FsRtl. 0x14 The ECP list has flags set that are illegal for a parameter list passed by a create caller. 0x15 The ECP list passed by the create caller is empty.
0x0000010D	WDF_VIOLATION	This indicates that Kernel-Mode Driver Framework (KMDF) detected that Windows found an error in a framework-based driver.	Parameter 1 indicates the specific error code of the bug check. 0x1 A framework-based driver has timed out during a

			power operation. This typically means that the device stack did not set the DO_POWER_PAGABLE bit and a driver attempted a pageable operation after the paging device stack was powered down. 0x2 An attempt is being made to acquire a lock that is currently being held. 0x3 Windows Driver Framework Verifier has encountered a fatal error. In particular, an I/O request was completed, but a framework request object cannot be deleted because there are outstanding references to the input buffer, the output buffer, or both. 0x4 A NULL parameter was passed to a function that required a non-NULL value. 0x5 A framework object handle of the incorrect type was passed to a framework object method. 0x6 A fatal error was made in handling a WDF request 0x7 A driver attempted to delete a framework object incorrectly by calling WdfObjectDereference to delete a handle instead of calling WdfObjectDelete. 0x8 An operation occurred on a DMA transaction object while it was not in the correct state. 0xA A fatal error has occurred while processing a request that is currently in the queue. 0xB An attempt to acquire or release a lock was invalid. 0xC A new state-changing PnP IRP arrived while the driver was processing another state-changing PnP IRP. 0xD A device's power policy owner received a power IRP that it did not request. There might be multiple power policy owners, but only one is allowed. A KMDF driver can change power policy ownership by calling WdfDeviceInitSetPowerPolicyOwnership. 0xE An event callback function did not return at the same IRQL at which it was called. The callback function changed the IRQL directly or indirectly (for example, by acquiring a spinlock, which raises IRQL to DISPATCH_LEVEL, but not releasing the spinlock). 0xF An event callback function entered a critical region, but it did not leave the critical region before returning.
0x0000010E	VIDEO_MEMORY_MANAGEMENT_INTERNAL	This indicates that the video memory manager has encountered a condition that it is unable to recover from.	This bug check is usually caused by a video driver behaving improperly. Parameter 1 is the only parameter of interest; this identifies the exact violation. 0x1 An attempt was made to rotate a non-rotate range. 0x2 An attempt was made to destroy a non-empty process heap. 0x3 An attempt to unmap from an aperture segment failed. 0x4 A rotation in a must-succeed path failed. 0x5 A deferred command failed. 0x6 An attempt was made to reallocate resources for an allocation that was having its eviction canceled. 0x7 An invalid attempt was made to defer free usage. 0x8 The split direct memory access (DMA) buffer contains an invalid reference. 0x9 An attempt to evict an allocation failed. 0xA An invalid attempt to use a pinned allocation was made. 0xB A driver returned an invalid error code from BuildPagingBuffer. 0xC A resource leak was detected in a segment. 0xD A segment is being used improperly. 0xE An attempt to map an allocation into an aperture segment failed. 0xF A driver returned an invalid error code from AcquireSwizzlingRange. 0x10 A driver returned an invalid error code from ReleaseSwizzlingRange. 0x11 An invalid attempt to use an aperture segment was made. 0x12 A driver overflowed the provided DMA buffer. 0x13 A driver overflowed the provided private data buffer. 0x14 An attempt to purge all segments failed. 0x15 An attempt was made to free a virtual address descriptor (VAD) that was still in the rotated state 0x16 A driver broke the guaranteed DMA buffer model contract.

			0x17 An unexpected system command failure occurred. 0x18 An attempt to release a pinned allocation's resource failed. 0x19 A driver failed to patch a DMA buffer. 0x1A The owner of a shared allocation was freed. 0x1B An attempt was made to release an aperture range that is still in use. 0x25 The GPU attempted to write over an undefined area of the aperture.
0x0000010F	RESOURCE_MANAGER_EXCEPTION_NOT_HANDLED	This indicates that the kernel transaction manager detected that a kernel-mode resource manager has raised an exception in response to a direct call-back.	The resource manager is in an unexpected and unrecoverable state.
0x00000111	RECURSIVE_NMI	This bug check indicates that a non-maskable-interrupt (NMI) occurred while a previous NMI was in progress.	This bug check occurs when there is an error in the system management interrupt (SMI) code, and an SMI interrupts an NMI and enables interrupts. Execution then continues with NMIs enabled, and another NMI interrupts the NMI in progress.
0x00000112	MSRPC_STATE_VIOLATION	This indicates that the Msrpc.sys driver has initiated a bug check.	The most common cause of this bug check is that the caller of the Msrpc.sys driver violated the state semantics for such a call. 0x01 A non-continuable exception was continued by the caller. 0x02 The advanced local procedure call (ALPC) returned an invalid error. 0x03 The caller unloaded the Microsoft remote procedure call (MSRPC) driver while it was still in use. It is likely that open binding handles remain. 0x04 0x05 An invalid close command was received from the ALPC. 0x06 An attempt was made to bind a remote procedure call (RPC) handle a second time. 0x07 An attempt was made to perform an operation on a binding handle that was not bound. 0x08 An attempt was made to set security information on a binding handle that was already bound. 0x09 An attempt was made to set an option on a binding handle that was already bound. 0x0A An attempt was made to cancel an invalid asynchronous remote procedure call. 0x0B An attempt was made to push on an asynchronous pipe call when it was not expected. 0x0C 0x0E An attempt was made to push on an asynchronous pipe without waiting for notification. 0x0F An attempt was made to synchronously terminate a pipe a second time. 0x15 An RPC internal error occurred. 0x16 Two causally ordered calls were issued in an order that cannot be enforced by the RPC. 0x17 A server manager routine did not unsubscribe from notifications prior to completing the call. 0x18 An invalid operation on the asynchronous handle occurred.
0x00000113	VIDEO_DXGKRNL_FATAL_ERROR	This indicates that the Microsoft DirectX graphics kernel subsystem has detected a violation.	This bug check is usually caused by a video driver behaving improperly.
0x00000114	VIDEO_SHADOW_DRIVER_FATAL_ERROR	This indicates that the shadow driver has detected a violation.	This bug check is usually caused by a video driver behaving improperly.
0x00000115	AGP_INTERNAL	This indicates that the accelerated graphics port (AGP) driver has detected a violation.	This bug check is usually caused by a video driver behaving improperly.
0x00000116	VIDEO_TDR_ERROR	This indicates that an attempt to reset the display driver and recover from a timeout failed.	This bug check is usually caused by a video driver behaving improperly.
0x00000117	VIDEO_TDR_TIMEOUT_DETECTED	This indicates that the display driver failed to respond in a timely fashion.	This bug check is usually caused by a video driver behaving improperly.
0x00000119	VIDEO_SCHEDULER_INTERNAL_ERROR	This indicates that the video scheduler has detected a fatal violation.	This bug check is usually caused by a video driver behaving improperly. Parameter 1 is the only parameter of interest and identifies the exact violation. 0x1 The driver has reported an invalid fence ID. 0x2 The driver failed upon the submission of a command. 0x3 The driver failed upon patching the command buffer. 0x4 The driver reported an invalid flip capability.

0x0000011A	EM_INITIALIZATION_FAILURE	Cannot initialize Evaluation Module connectors correctly.	This bug check is usually caused by a video driver behaving improperly.
0x0000011B	DRIVER_RETURNED_HOLDING_CANCEL_LOCK	This bug check indicates that a driver has returned from a cancel routine that holds the global cancel lock. This causes all later cancellation calls to fail, and results in either a deadlock or another bug check.	The driver calls the IoCancelIrpIoCancelIrp function to cancel an individual I/O request packet (IRP). This function acquires the cancel spin lock, sets the cancel flag in the IRP, and then calls the cancel routine specified by the appropriate field in the IRP, if a routine was specified. The cancel routine is expected to release the cancel spin lock. If there is no cancel routine, the cancel spin lock is released.
0x0000011C	ATTEMPTED_WRITE_TO_CM_PROTECTED_STORAGE	This bug check indicates that an attempt was made to write to the read-only protected storage of the configuration manager.	When it is possible, the name of the driver that is attempting the write operation is printed as a Unicode string on the bug check screen and then saved in KiBugCheckDriver.
0x0000011D	EVENT_TRACING_FATAL_ERROR	This bug check indicates that the Event Tracing subsystem has encountered an unexpected fatal error.	See general suggestions above the table.
0x00000121	DRIVER_VIOLATION	This bug check indicates that a driver has caused a violation.	Use a kernel debugger and view the call stack to determine the name of the driver that caused the violation.
0x00000122	WHEA_INTERNAL_ERROR	This bug check indicates that an internal error in the Windows Hardware Error Architecture (WHEA) has occurred.	Errors can result from a bug in the implementation of a platform-specific hardware error driver (PSHED) plug-in supplied by a vendor, the firmware implementation of error records, or the firmware implementation of error injection. First parameter gives more details about exception. 0x1 Failed to allocate enough memory for all the error sources in the hardware error source table. 0x2 Failed to allocate enough memory for a WHEA information block for each processor. 0x5 WHEA failed to allocate enough memory for the error sources, or the error source enumeration failed. 0x6 Failed to initialize the error source (Parameter 4) during the phase specified by Parameter 3. 0x7 Failed to allocate enough memory. 0x8 Failed to allocate enough memory for all the error source descriptors. 0x9 WHEA received an uncorrected error source from an invalid error source. 0xA Failed to allocate an error record for an uncorrected error. 0xB Failed to populate the error record for an uncorrected error.
0x00000124	WHEA_UNCORRECTABLE_ERROR	This bug check indicates that a page that should have been filled with zeros was not.	This bug check might occur because of a hardware error or because a privileged component of the operating system modified a page after freeing it.
0x00000127	PAGE_NOT_ZERO	This bug check indicates that a single-bit error was found in this page. This is a hardware memory error.	See general suggestions above the table.
0x0000012B	FAULTY_HARDWARE_CORRUPTED_PAGE	This bug check indicates that a single-bit error was found in this page. This is a hardware memory error.	See general suggestions above the table.
0x0000012C	EXFAT_FILE_SYSTEM	This bug check indicates that a problem occurred in the Extended File Allocation Table (exFAT) file system.	This bug check is caused by the file system as a last resort when its internal accounting is in an unsupportable state and to continue poses a large risk of data loss. The file system never causes this bug check when the on disk structures are corrupted, the disk sectors go bad, or a memory allocation fails. Bad sectors could lead to a bug check, for example, when a page fault occurs in kernel code or data and the memory manager cannot read the pages. However, for this bug check, the file system is not the cause.
0x00000133	DPC_WATCHDOG_VIOLATION	This bug check indicates that the DPC watchdog executed, either because it detected a single long-running deferred procedure call (DPC), or because the system spent a prolonged time at an interrupt request level (IRQL) of DISPATCH_LEVEL or above.	The value of Parameter 1 indicates whether a single DPC exceeded a timeout, or whether the system cumulatively spent an extended period of time at IRQL DISPATCH_LEVEL or above. The first parameter gives more details about the exception. 0 A single DPC or ISR exceeded its time allotment. The offending component can usually be identified with a stack trace. 1 The system cumulatively spent an extended period of time at IRQL DISPATCH_LEVEL or above. The offending component can usually be identified with a stack trace.
0x00000138	GPIO_CONTROLLER_DRIVER_ERROR	This bug check indicates that the GPIO class extension driver encountered a fatal error.	The cause of Error depends on the value of Parameter 1. 1 The GPIO controller managing the specific GSIV is not registered.

			2 A client driver specified an invalid context to a lock or unlock request. 3 PoFx requested that the GPIO controller send a bank through an inappropriate F1 power state and/or a critical transition. 4 PoFx requested that the GPIO controller send a bank through an inappropriate F0 power state and/or a critical transition. 5 An on-Soc GPIO interrupt operation failed. 6 An on-Soc GPIO IO operation failed. 7 A _DSM method returned malformed data.
0x00000139	KERNEL_SECURITY_CHECK_FAILURE	This bug check indicates that the kernel has detected the corruption of a critical data structure.	The cause of Error depends on the value of Parameter 1. 0 A stack-based buffer has been overrun (legacy /GS violation). 1 VTGuard instrumentation code detected an attempt to use an illegal virtual function table. Typically, a C++ object was corrupted, and then a virtual method call was attempted using the corrupted object's this pointer. 2 Stack cookie instrumentation code detected a stack-based buffer overrun (/GS violation). 3 A LIST_ENTRY was corrupted (for example, a double remove). For more information, see the following Cause section. This type of bug check can be difficult to track down and indicates that an inconsistency has been introduced into a doubly-linked list (detected when an individual list entry element is added to or removed from the list). Unfortunately, the inconsistency is not necessarily detected at the time when the corruption occurred, so some detective work may be necessary to identify the root cause. 5 An invalid parameter was passed to a function that considers invalid parameters fatal. 6 The stack cookie security cookie was not properly initialized by the loader. This may be caused by building a driver to run only on Windows 8 and attempting to load the driver image on an earlier version of Windows. To avoid this problem, you must build the driver to run on an earlier version of Windows. 7 A fatal program exit was requested. 8 A array bounds check inserted by the compiler detected an illegal array indexing operation. 9 A call to RtlQueryRegistryValues was made specifying RTL_QUERY_REGISTRY_DIRECT without RTL_QUERY_REGISTRY_TYPECHECK, and the target value was not in a trusted system hive.
0x00000144	BUGCODE_USB3_DRIVER	An error with USB 3 driver occurred.	The cause of Error depends on the value of Parameter 1. 0x1 A client driver used an URB that it had previously sent to the core stack. 0x2 A boot or paging device failed re-enumeration. 0x3 A client driver sent a corrupted URB to the core stack. This can happen because the client driver did not allocate the URB using USBD_xxxUrbAllocate or because the client driver did a buffer underrun for the URB. 0x800 An Open Static Streams request was sent at IRQL > PASSIVE LEVEL. 0x801 A client driver attempted to open static streams before querying for streams capability. A client driver cannot open a static stream until after it successfully queries for the streams capability. For more information, see Remarks. 0x802 A Client driver tried to open an invalid number of static streams. The number of streams cannot be 0 and cannot be greater than the value returned to the client driver in the query USB capability call. 0x803 A client driver attempted to open static streams for an endpoint that already had static streams open. Before opening static streams, the client driver must close the previously opened static streams. 0x804 A client driver forgot to close a handle it created earlier using USBD_CreateHandle or forgot to free an URB it allocated. 0x805 A client driver sent a Close Static Streams URB in an invalid state (for example, after processing D0 Exit). 0x806 A client driver attempted to send a chained MDL before querying for chained MDL capability. A client driver cannot send a chained MDL until after it

			successfully queries for the chained MDL capability. For more information, see Remarks. 0x807 A client driver sent an URB to the core stack with a transfer buffer length longer than the byte count (returned by MmGetMdlByteCount) of the MDL passed in. For more information, see Remarks.
0x0000014B	SOC_SUBSYSTEM_FAILURE	This indicates that an unrecoverable error was encountered in a System on a Chip (SoC) subsystem.	See general suggestions above the table.
0x1000007E	SYSTEM_THREAD_EXCEPTION_NOT_HANDLED_M	This indicates that a system thread generated an exception which the error handler did not catch.	Has the same meaning and parameters as bug check 0x7E SYSTEM_THREAD_EXCEPTION_NOT_HANDLED
0x1000007F	UNEXPECTED_KERNEL_MODE_TRAP_M	This indicates that a trap was generated by the Intel CPU and the kernel failed to catch this trap.	Has the same meaning and parameters as bug check 0x7F UNEXPECTED_KERNEL_MODE_TRAP
0x1000008E	KERNEL_MODE_EXCEPTION_NOT_HANDLED_M	This indicates that a kernel-mode program generated an exception which the error handler did not catch.	Has the same meaning and parameters as bug check 0x8E KERNEL_MODE_EXCEPTION_NOT_HANDLED
0x100000EA	THREAD_STUCK_IN_DEVICE_DRIVER_M	This indicates that a thread in a device driver is endlessly spinning.	Has the same meaning and parameters as bug check 0xEA THREAD_STUCK_IN_DEVICE_DRIVER
0xC0000218	STATUS_CANNOT_LOAD_REGISTRY_FILE	This indicates that a registry file could not be loaded.	This error occurs if a necessary registry hive file cannot be loaded. Usually this means the file is corrupt or is missing. The name of the damaged file is displayed as part of the message. In rare instances, this error can be caused by a driver that has corrupted the registry image in memory, or by a memory error in this region.
0xC000021A	STATUS_SYSTEM_PROCESS_TERMINATED	This means that an error has occurred in a crucial user-mode subsystem.	This error occurs when a user-mode subsystem, such as WinLogon or the Client Server Run-Time Subsystem (CSRSS), has been fatally compromised and security can no longer be guaranteed. In response, the operating system switches to kernel mode. Microsoft Windows cannot run without WinLogon or CSRSS. Therefore, this is one of the few cases where the failure of a user-mode service can shut down the system. Running the kernel debugger is not useful in this situation because the actual error occurred in a user-mode process. Resolving an error in a user-mode device driver, system service, or third-party application: Because bug check 0xC000021A occurs in a user-mode process, the most common culprits are third-party applications, updated device driver, system service, or third-party application. The new software should be removed or disabled. Mismatched system files can also cause this error. This can occur if you have restored your hard disk from a backup. Some backup programs might skip restoring system files that they determine are in use.
0xC0000221	STATUS_IMAGE_CHECKSUM_MISMATCH	This indicates that a driver or a system DLL has been corrupted.	This bug check results from a serious error in a driver or other system file. The file header checksum does not match the expected checksum. This can also be caused by faulty hardware in the I/O path to the file (a disk error, faulty RAM, or a corrupted page file).
0xDEADDEAD	MANUALLY_INITIATED_CRASH1	This indicates that the user deliberately initiated a crash dump from either the kernel debugger or the keyboard.	This is not a real error. See Force A System Crash using Keyboard section.