Contents

Windows Device Console (Devcon.exe)
http://msdn.microsoft.com/en-us/library/windows/hardware/ff544707%28v=vs.85%29.aspx

DevCon (Devcon.exe), the Device Console, is a command-line tool that displays detailed information about devices on computers running Windows. You can use DevCon to enable, disable, install, configure, and remove devices.

DevCon runs on Microsoft Windows 2000 and later versions of Windows.

Where can I download DevCon?

DevCon (Devcon.exe) is included when you install the WDK, Visual Studio, and the Windows SDK for desktop apps. For information about downloading the kits, see Windows Hardware Downloads.
Windows Driver Kit (WDK) 8 and Windows Driver Kit (WDK) 8.1 (installation path)
%WindowsSdkDir%\tools\x64\devcon.exe
%WindowsSdkDir%\tools\x86\devcon.exe
%WindowsSdkDir%\tools\arm\devcon.exe
Note The Visual Studio environment variable, %WindowsSdkDir%, represents the path to the Windows kits directory where the kits are installed, for example, C:\Program Files (x86)\Windows Kits\8.1.

This section includes:
DevCon Commands
DevCon Examples

What you can do with DevCon

Windows driver developers and testers can use DevCon to verify that a driver is installed and configured correctly, including the proper INF files, driver stack, driver files, and driver package. You can also use the DevCon commands (enable, disable, install, start, stop, and continue) in scripts to test the driver.
DevCon is a command-line tool that performs device management functions on local computers and remote computers.

Note To run DevCon commands on a remote computer, the Group Policy setting must allow the Plug and Play service to run on the remote computer. On computers that run Windows Vista and Windows 7, the Group Policy disables remote access to the service by default. On computers that run WDK 8.1 and WDK 8, the remote access is unavailable.

Devcon features include:

- **Display driver and device info** DevCon can display the following properties of drivers and devices on local computers, and remote computers (running Windows XP and earlier):
  - Hardware IDs, compatible IDs, and device instance IDs. These identifiers are described in detail in Device Identification Strings.
  - Device setup classes
  - The devices in a device setup class
  - INF files and device driver files
  - Details of driver packages
  - Hardware resources
  - Device status
  - Expected driver stack
  - Third-party driver packages in the driver store
- **Search for devices** DevCon can search for installed and uninstalled devices on a local or remote computer by hardware ID, device instance ID, or device setup class.
- **Change device settings** DevCon can change the status or configuration of Plug and Play (PnP) devices on the local computer in the following ways:
  - Enable a device
  - Disable a device
  - Update drivers (interactive and noninteractive)
  - Install a device (create a devnode and install software)

    - Remove a device from the device tree and delete its device stack
    - Rescan for Plug and Play devices
    - Add, delete, and reorder the hardware IDs of root-enumerated devices
    - Change the upper and lower filter drivers for a device setup class
    - Add and delete third-party driver packages from the driver store
- **Restart the device or computer** DevCon can restart a local device, reboot the local system on demand, or reboot the local system if required for another DevCon operation.

DevCon source code

The DevCon source code is also available so that you can examine the methods that DevCon uses to retrieve and change setup and configuration data. DevCon illustrates the use of general setup functions, device installation functions, and PnP Configuration Manager functions. The source code for the Device Console (DevCon) Tool is available from the MSDN Hardware Samples code gallery.

Device Console (DevCon.exe) Commands

http://msdn.microsoft.com/en-us/library/windows/hardware/ff544766%28v=vs.85%29.aspx

Device Console (DevCon.exe) Examples

http://msdn.microsoft.com/en-us/library/windows/hardware/ff544746%28v=vs.85%29.aspx

This section provides examples of the following Device Console (DevCon.exe) commands:

DevCon HwIDs

Example 1: Find all hardware IDs
Example 2: Find hardware IDs by using a pattern
Example 3: Find hardware IDs by using a class

DevCon Classes

Example 4: List classes on the local computer
Example 5: List classes on the remote computer

DevCon ListClass

Example 6: List the devices in a device setup class
Example 7: List the devices in multiple classes on a remote computer

DevCon DriverFiles

Example 8: List all driver files
Example 9: List the driver files of a particular device

DevCon DriverNodes

Example 10: List driver packages by hardware ID pattern
Example 11: List driver packages by device instance ID pattern

DevCon Resources

Example 12: List resources of a class of devices
Example 13: List resources of device on a remote computer by ID

DevCon Stack

Example 14: Display the driver stack for storage devices
Example 15: Find the setup class of a device
Example 16: Display the stack for related devices on a remote computer

DevCon Status

Example 17: Display the status of all devices on the local computer
Example 18: Display the status of a device by device instance ID
Example 19: Display the status of related devices on a remote computer

DevCon Find

Example 20: Find devices by hardware ID pattern
Example 21: Find devices by device instance ID or class

DevCon FindAll

Example 22: Find (and find all) devices in a setup class

DevCon ClassFilter

Example 1: Find all hardware IDs

Because DevCon operations use IDs and ID patterns to identify devices, a common first step in using DevCon is to create a hardware ID reference file for devices on the computer.

The following command uses the **DevCon HwIDs** operation, which returns the IDs and the device description. It uses the wildcard character (*****) to represent all devices on the local computer.

```
devcon hwids *
```

Because the output is lengthy and used repeatedly, save the output in a text file for reference.

The following command uses the wildcard character (*****) to represent all devices on the computer. It uses the redirection character (**>**) to save the command output in the hwids.txt file.

```
devcon hwids * > hwids.txt
```

The following command finds the hardware IDs of devices on a remote computer, Server01. It uses the **/m** parameter to specify the name of the remote computer. The command redirects the output to the server01_hwids.txt file for later reference.

Note This command fails unless the user has the required permissions on the remote computer. To run DevCon commands on a remote computer, the Group Policy setting must allow the Plug and Play service to run on the remote computer. On computers that run Windows Vista and Windows 7, the Group Policy disables remote access to the service by default. On computers that run Windows Driver Kit (WDK) 8.1 and Windows Driver Kit (WDK) 8, the remote access is unavailable.

```
devcon /m:\\server01 hwids * > server01_hwids.txt
```

Example 2: Find hardware IDs by using a pattern
To find the hardware IDs of a particular device, enter the hardware ID or pattern, the compatible ID or pattern, the device instance ID or pattern, or the name of the device setup class.
The following command uses the **DevCon HwIDs** operation and a pattern to find the hardware IDs of the floppy disk drive on the computer. (The user assumes that the pattern appears in one of the device identifiers.) The command uses the wildcard character (*****) to represent all characters that might precede or follow the word "floppy" in any of the IDs.

```
devcon hwids *floppy*
```

In response, DevCon displays the device instance ID, hardware ID, and compatible ID of the floppy disk drive on the computer. You can use these IDs in subsequent DevCon commands.

```
FDC\GENERIC_FLOPPY_DRIVE\5&39194F6D&0&0
   Name: Floppy disk drive
   Hardware ID's:
      FDC\GENERIC_FLOPPY_DRIVE
   Compatible ID's:
      GenFloppyDisk
1 matching device(s) found.
```

In this case, the phrase "floppy" occurs in the hardware ID or compatible ID of only one device on the computer. If it occurs in the ID of more than one device, all devices with "floppy" in their IDs appear in the output.
Example 3: Find hardware IDs by using a class
The following command uses the **DevCon HwIDs** operation and a device setup class to find the hardware IDs of all devices in the Ports device setup class. The equal sign (**=**) preceding the class name indicates that it is a class, not an ID.

```
devcon hwids =ports
```

In response, DevCon displays the hardware IDs and compatible IDs of the three devices in the Ports setup class.

```
ACPI\PNP0401\4&B4063F4&0
   Name: ECP Printer Port (LPT1)
   Hardware ID's:
      ACPI\PNP0401
      *PNP0401
ACPI\PNP0501\1
   Name: Communications Port (COM1)
   Hardware ID's:
      ACPI\PNP0501
      *PNP0501
ACPI\PNP0501\2
   Name: Communications Port (COM2)
   Hardware ID's:
      ACPI\PNP0501
      *PNP0501
3 matching device(s) found.
```

Example 4: List classes on the local computer
Because DevCon operations can use the device setup class to identify devices, it is useful to create a reference file of the device setup classes of devices on the computer.
The following command uses the **DevCon Classes** operation, which returns a list and description of all classes on the computer.

```
devcon classes
```

Because the output is lengthy and used repeatedly, save the output in a text file for reference.
The following command displays all device classes on the computer. It uses the redirection character (**>**) to save the command output in the classes.txt file.

```
devcon classes > classes.txt
```

Example 5: List classes on the remote computer
The following command uses the **DevCon Classes** operation to list the device setup classes on a remote computer, Server01:

```
devcon /m:\\server01 classes
```

Because the output is lengthy and used repeatedly, save the output in a text file for reference.
The following command uses the redirection character (**>**) to save the command output in the server01_classes.txt file.

```
devcon /m:\\server01 classes > server01_classes.txt
```

Example 6: List the devices in a device setup class
The following command uses the **DevCon ListClass** operation to list the devices in Net, the device setup class for network adapters.

```
devcon listclass net
```

In response, DevCon displays the device instance ID and description of each device in the Net setup class.

```
Listing 6 device(s) for setup class "Net" (Network adapters).
PCI\VEN_10B7&DEV_9200&SUBSYS_00BE1028&REV_78\4&BB7B4AE&0&60F0: 3Com 3C920 Integrated Fast Ethernet Controller (3C905C-TX Compatible)
ROOT\MS_L2TPMINIPORT\0000                  : WAN Miniport (L2TP)
ROOT\MS_NDISWANIP\0000                    : WAN Miniport (IP)
ROOT\MS_PPPOEMINIPORT\0000                  : WAN Miniport (PPPOE)
ROOT\MS_PPTPMINIPORT\0000                  : WAN Miniport (PPTP)
ROOT\MS_PTIMINIPORT\0000                   : Direct Parallel
```

This display, although interesting, does not provide the hardware IDs of the devices in the Net setup class. The following command uses the **DevCon HwIDs** operation to list the devices in the Net setup class. In a **DevCon HwIDs** command, the class name is preceded by an equal sign (**=**) to indicate that it is a class, not an ID.

```
devcon hwids =net
```

The resulting display lists the devices in the Net class and includes the device instance ID, hardware IDs, and compatible IDs of devices in the class.

```
PCI\VEN_10B7&DEV_9200&SUBSYS_00BE1028&REV_78\4&BB7B4AE&0&60F0
    Name: 3Com 3C920 Integrated Fast Ethernet Controller (3C905C-TX Compatible)
    Hardware ID's:
        PCI\VEN_10B7&DEV_9200&SUBSYS_00BE1028&REV_78
        PCI\VEN_10B7&DEV_9200&SUBSYS_00BE1028
        PCI\VEN_10B7&DEV_9200&CC_020000
        PCI\VEN_10B7&DEV_9200&CC_0200
    Compatible ID's:
        PCI\VEN_10B7&DEV_9200&REV_78
        PCI\VEN_10B7&DEV_9200
        PCI\VEN_10B7&CC_020000
        PCI\VEN_10B7&CC_0200
 PCI\VEN_10B7
        PCI\CC_020000
```

PCI\CC_0200
ROOT\MS_L2TPMINIPORT\0000
   Name: WAN Miniport (L2TP)
   Hardware ID's:
      ms_l2tpminiport
ROOT\MS_NDISWANIP\0000
   Name: WAN Miniport (IP)
   Hardware ID's:
      ms_ndiswanip
ROOT\MS_PPPOEMINIPORT\0000
   Name: WAN Miniport (PPPOE)
   Hardware ID's:
      ms_pppoeminiport
ROOT\MS_PPTPMINIPORT\0000
   Name: WAN Miniport (PPTP)
   Hardware ID's:
      ms_pptpminiport
ROOT\MS_PTIMINIPORT\0000
   Name: Direct Parallel
   Hardware ID's:
      ms_ptiminiport
6 matching device(s) found.

Example 7: List the devices in multiple classes on a remote computer
The following command uses the **DevCon ListClass** operation to list the devices in the DiskDrive, CDROM, and TapeDrive classes on Server01, a remote computer.

devcon /m:\\server01 listclass diskdrive cdrom tapedrive

In response, DevCon displays the devices in those classes on the remote computer.

Listing 1 device(s) for setup class "DiskDrive" (Disk drives) on \\server01.
IDE\DISKWDC_WD204BA_____16.13M16\4457572D414D373032313633393820312
0202020: WDC WD204BA
Listing 1 device(s) for setup class "CDROM" (DVD/CD-ROM drives) on \\server01.
IDE\CDROMSAMSUNG_DVD-ROM_SD-608_____2.2_____\4&13B4AFD&0&0.0.0: SAMSUNG DVD-ROM SD-608
No devices for setup class "TapeDrive" (Tape drives) on \\server01.

Example 8: List all driver files
The following command uses the **DevCon DriverFiles** operation to list the file names of drivers that devices on the system use. The command uses the wildcard character (*****) to indicate all devices on the system. Because the output is extensive, the command uses the redirection character (**>**) to redirect the output to a reference file, driverfiles.txt.

devcon driverfiles * > driverfiles.txt

Example 9: List the driver files of a particular device
The following command uses the **DevCon DriverFiles** operation to search for the device driver that the mouse device on the local computer uses. It identifies the device by one of its hardware IDs, HID\Vid_045e&Pid_0039&Rev_0121. The hardware ID is enclosed in quotation marks because it includes the ampersand character (**&**).

devcon driverfiles "HID\Vid_045e&Pid_0039&Rev_0121"

In response, DevCon displays the two device drivers that support the mouse device.

HID\VID_045E&PID_0039\6&DC36FDE&0&0000

Name: Microsoft USB IntelliMouse Optical
Driver installed from c:\windows\inf\msmouse.inf [HID_Mouse_Inst]. 2 file(s)
used by driver:
C:\WINDOWS\System32\DRIVERS\mouhid.sys
C:\WINDOWS\System32\DRIVERS\mouclass.sys
1 matching device(s) found.

Example 10: List driver packages by hardware ID pattern

The following command uses the **DevCon DriverNodes** command and an ID pattern to list the driver nodes of software-enumerated devices. Patterns are useful for finding information about similar devices that might not be in the same setup class.

The following command uses the ID pattern **sw*** to specify devices whose hardware IDs or compatible IDs begin with "sw," that is, software-enumerated devices.

```
devcon drivernodes sw*
```

In response, DevCon displays the driver nodes of software-enumerated devices on the system.

```
SW\{A7C7A5B0-5AF3-11D1-9CED-00A024BF0407}\{9B365890-165F-11D0-A195-0020AFD156E4}

Name: Microsoft Kernel System Audio Device
DriverNode #0:
Inf file is c:\windows\inf\wdmaudio.inf
Inf section is WDM_SYSAUDIO
Driver description is Microsoft Kernel System Audio Device
Manufacturer name is Microsoft
Provider name is Microsoft
Driver date is 7/1/2001
Driver version is 5.1.2535.0
Driver node rank is 0
Driver node flags are 00002244
Inf is digitally signed
SW\{B7EAFDC0-A680-11D0-96D8-00AA0051E51D}\{9B365890-165F-11D0-A195-0020AFD156E4}

Name: Microsoft Kernel Wave Audio Mixer
DriverNode #0:
Inf file is c:\windows\inf\wdmaudio.inf
Inf section is WDM_KMIXER
Driver description is Microsoft Kernel Wave Audio Mixer
Manufacturer name is Microsoft
Provider name is Microsoft
Driver date is 7/1/2001
Driver version is 5.1.2535.0
Driver node rank is 0
Driver node flags are 00002244
Inf is digitally signed
SW\{CD171DE3-69E5-11D2-B56D-0000F8754380}\{9B365890-165F-11D0-A195-0020AFD156E4}

Name: Microsoft WINMM WDM Audio Compatibility Driver
DriverNode #0:
Inf file is c:\windows\inf\wdmaudio.inf
Inf section is WDM_WDMAUD
Driver description is Microsoft WINMM WDM Audio Compatibility Driver
Manufacturer name is Microsoft
Provider name is Microsoft
Driver date is 7/1/2001
Driver version is 5.1.2535.0
Driver node rank is 0
```

```
    Driver node flags are 00002244
      Inf is digitally signed
3 matching device(s) found.
```

Example 11: List driver packages by device instance ID pattern

The following command uses the **DevCon DriverNodes** operation to list the driver packages of all devices whose device instance IDs begin with ROOT\MEDIA, that is, devices in the Enum\Root\Media registry subkey. The command uses the at character (**@**) to indicate that the phrase is in the device instance ID.

```
devcon drivernodes @ROOT\MEDIA*
```

In response, DevCon displays the driver nodes of devices whose device instance ID begins with "ROOT\MEDIA."

```
ROOT\MEDIA\MS_MMACM
    Name: Audio Codecs
DriverNode #0:
    Inf file is c:\windows\inf\wave.inf
    Inf section is MS_MMACM
    Driver description is Audio Codecs
    Manufacturer name is (Standard system devices)
    Provider name is Microsoft
    Driver date is 7/1/2001
    Driver version is 5.1.2535.0
    Driver node rank is 0
    Driver node flags are 00002240
      Inf is digitally signed
ROOT\MEDIA\MS_MMDRV
    Name: Legacy Audio Drivers
DriverNode #0:
    Inf file is c:\windows\inf\wave.inf
    Inf section is MS_MMDRV
    Driver description is Legacy Audio Drivers
    Manufacturer name is (Standard system devices)
    Provider name is Microsoft
    Driver date is 7/1/2001
    Driver version is 5.1.2535.0
    Driver node rank is 0
    Driver node flags are 00002240
      Inf is digitally signed
ROOT\MEDIA\MS_MMMCI
    Name: Media Control Devices
DriverNode #0:
    Inf file is c:\windows\inf\wave.inf
    Inf section is MS_MMMCI
    Driver description is Media Control Devices
    Manufacturer name is (Standard system devices)
    Provider name is Microsoft
    Driver date is 7/1/2001
    Driver version is 5.1.2535.0
    Driver node rank is 0
    Driver node flags are 00002240
      Inf is digitally signed
ROOT\MEDIA\MS_MMVCD
    Name: Legacy Video Capture Devices
DriverNode #0:
    Inf file is c:\windows\inf\wave.inf
    Inf section is MS_MMVCD
    Driver description is Legacy Video Capture Devices
```

```
    Manufacturer name is (Standard system devices)
    Provider name is Microsoft
    Driver date is 7/1/2001
    Driver version is 5.1.2535.0
    Driver node rank is 0
    Driver node flags are 00002240
      Inf is digitally signed
ROOT\MEDIA\MS_MMVID
    Name: Video Codecs
DriverNode #0:
    Inf file is c:\windows\inf\wave.inf
    Inf section is MS_MMVID
    Driver description is Video Codecs
    Manufacturer name is (Standard system devices)
    Provider name is Microsoft
    Driver date is 7/1/2001
    Driver version is 5.1.2535.0
    Driver node rank is 0
    Driver node flags are 00002240
      Inf is digitally signed
5 matching device(s) found.
```

Example 12: List resources of a class of devices

The following command uses the **DevCon Resources** operation to display the resources allocated to devices in the Hdc device setup class. This class includes IDE controllers. The equal sign (**=**) is prepended to "hdc" to indicate that it is a class and not an ID.

```
devcon resources =hdc
```

In response, DevCon lists the resources allocated to IDE controllers on the local computer.

```
PCI\VEN_8086&DEV_244B&SUBSYS_00000000&REV_02\3&29E81982&0&F9
    Name: Intel(r) 82801BA Bus Master IDE Controller
    Device is currently using the following resources:
      IO  : ffa0-ffaf
PCIIDE\IDECHANNEL\4&37E53584&0&0
    Name: Primary IDE Channel
    Device is currently using the following resources:
      IO  : 01f0-01f7
      IO  : 03f6-03f6
      IRQ : 14
PCIIDE\IDECHANNEL\4&37E53584&0&1
    Name: Secondary IDE Channel
    Device is currently using the following resources:
      IO  : 0170-0177
      IO  : 0376-0376
      IRQ : 15
3 matching device(s) found.
```

Example 13: List resources of device on a remote computer by ID

The following command uses the **DevCon Resources** operation to list the resources allocated to the system timer on Server01, a remote computer. The command uses the hardware ID of the system timer, ACPI\PNP0100, to specify the device.

```
devcon /m:\\Server01 resources *PNP0100
```

In response, DevCon displays the resources of the Server01 system timer.

```
ROOT\*PNP0100\PNPBIOS_8
   Name: System timer
   Device has the following resources reserved:
      IO  : 0040-005f
      IRQ : 0
1 matching device(s) found on \\server01.
```

The following command uses the device instance ID of the remote system timer in the DevCon resources command. The at character (**@**) indicates that the string is a device instance ID, not a hardware ID or compatible ID.

```
devcon /m:\\Server01 resources @ACPI\PNP0100\4&b4063f4&0
```

Example 14: Display the driver stack for storage devices

The following command uses the **DevCon Stack** operation to search for devices in the Volume setup class and display the expected driver stack for those devices. The equal sign (**=**) indicates that the string is a class name.

```
devcon stack =Volume
```

In response, DevCon displays the expected stack for the devices in the Volume class. The returned data includes the device instance ID and description of each device, the GUID and name of the device setup class, the names of upper and lower filter drivers, and controlling services (if any).

```
STORAGE\VOLUME\1&30A96598&0&SIGNATURE32323533OFFSET271167600LENGTH6E00D0C00
   Name: Generic volume
   Setup Class: {71A27CDD-812A-11D0-BEC7-08002BE2092F} Volume
   Class upper filters:
      VolSnap
   Controlling service:
      (none)
STORAGE\VOLUME\1&30A96598&0&SIGNATURE32323533OFFSET7E00LENGTH27115F800
   Name: Generic volume
   Setup Class: {71A27CDD-812A-11D0-BEC7-08002BE2092F} Volume
   Class upper filters:
      VolSnap
   Controlling service:
      (none)
2 matching device(s) found.
```

Example 15: Find the setup class of a device

The **DevCon Stack** operation returns the setup class of a device in addition to the upper and lower filter drivers. The following commands find the setup class of the printer port interface by finding its device instance ID and then using the device instance ID to find its setup class.

The following command uses the **DevCon HwIDs** operation to find the device instance ID of the printer port interface by using "LPT," a phrase in the printer port hardware ID.

```
devcon hwids *lpt*
```

In response, DevCon returns the device instance ID (displayed in bold text) and the hardware ID of the printer port interface.

```
LPTENUM\MICROSOFTRAWPORT\5&CA97D7E&0&LPT1
   Name: Printer Port Logical Interface
   Hardware ID's:
      LPTENUM\MicrosoftRawPort958A
      MicrosoftRawPort958A
1 matching device(s) found.
```

The next command uses the **DevCon Stack** operation to find the device setup class of the device represented by the device instance ID. An at character (**@**) identifies the ID as a device instance ID. The ID is enclosed in quotation marks because it includes ampersand characters.

```
devcon stack "@LPTENUM\MICROSOFTRAWPORT\5&CA97D7E&0&LPT1"
```

In response, DevCon displays the driver stack for the printer port interface, including the class. The display reveals that the printer port is in the System class.

```
LPTENUM\MICROSOFTRAWPORT\5&CA97D7E&0&LPT1
   Name: Printer Port Logical Interface
   Setup Class: {4D36E97D-E325-11CE-BFC1-08002BE10318} System
   Controlling service:
      (none)
1 matching device(s) found.
```

Example 16: Display the stack for related devices on a remote computer

The following command uses the **DevCon Stack** operation to display the expected stack for miniport driver devices on Server01, a remote computer. It searches for devices in the Net setup class that have "miniport" in their hardware ID or compatible ID.

Note that this command first limits the search to the Net setup class and then finds the "miniport" string. It does not find devices other than those in the Net setup class.

```
devcon /m:\\server01 stack =net *miniport*
```

In response, DevCon displays the expected stack for miniport drivers on Server01.

```
ROOT\MS_L2TPMINIPORT\0000
   Name: WAN Miniport (L2TP)
   Setup Class: {4D36E972-E325-11CE-BFC1-08002BE10318} Net
   Controlling service:
      Rasl2tp
ROOT\MS_PPPOEMINIPORT\0000
   Name: WAN Miniport (PPPOE)
   Setup Class: {4D36E972-E325-11CE-BFC1-08002BE10318} Net
   Controlling service:
      RasPppoe
   Lower filters:
      NdisTapi
ROOT\MS_PPTPMINIPORT\0000
   Name: WAN Miniport (PPTP)
   Setup Class: {4D36E972-E325-11CE-BFC1-08002BE10318} Net
   Controlling service:
      PptpMiniport
   Lower filters:
      NdisTapi
ROOT\MS_PTIMINIPORT\0000
   Name: Direct Parallel
   Setup Class: {4D36E972-E325-11CE-BFC1-08002BE10318} Net
   Controlling service:
      Raspti
   Lower filters:
      PtiLink
4 matching device(s) found on \\Server01.
```

Example 17: Display the status of all devices on the local computer

The following command uses the **DevCon Status** operation to find the status of all devices on the local computer. It then saves the status in the status.txt file for logging or later review. The command uses the wildcard character (*****) to represent all devices and the redirection character (**>**) to redirect the output to the status.txt file.

```
devcon status * > status.txt
```

Example 18: Display the status of a device by device instance ID

The most reliable way to find the status of a particular device is to use the device instance ID of the device. The following command uses the device instance ID of the I/O controller on the local computer in a **DevCon Status** command. The command includes the device instance ID of the device, PCI\VEN_8086&DEV_1130&SUBSYS_00000000&REV_02\3&29E81982&0&00. The at character (**@**) prefixed to the ID identifies the string as a device instance ID. The ID must be enclosed in quotation marks because it includes ampersand characters.

```
devcon status "@PCI\VEN_8086&DEV_1130&SUBSYS_00000000&REV_02\3&29E81982&0&00"
```

In response, DevCon displays the status of the I/O controller.

```
PCI\VEN_8086&DEV_1130&SUBSYS_00000000&REV_02\3&29E81982&0&00
   Name: Intel(R) 82815 Processor to I/O Controller - 1130
   Driver is running.
1 matching device(s) found.
```

Example 19: Display the status of related devices on a remote computer

The following command uses the **DevCon Status** operation to display the status of particular storage-related devices on Server01, a remote computer. It searches for the following devices:

- Disk drive, GenDisk
- CD-ROM drive, GenCdRom
- Floppy disk drive, FDC\GENERIC_FLOPPY_DRIVE
- Volumes, STORAGE\Volume
- Logical disk manager, ROOT\DMIO
- Volume manager, ROOT\FTDISK
- Floppy disk controller, ACPI\PNP0700

In the command, each ID is separated from the others by spaces. Note that GenDisk and GenCdRom are compatible IDs, whereas the other IDs are hardware IDs.

```
devcon /m:\\server01 status GenDisk GenCdRom FDC\GENERIC_FLOPPY_DRIVE STORAGE\Volume ROOT\DMIO
ROOT\FTDISK ACPI\PNP0700
```

In response, DevCon displays the status of each device.

```
FDC\GENERIC_FLOPPY_DRIVE\1&3A2146F1&0&0
   Name: Floppy disk drive
   Driver is running.
IDE\CDROMSAMSUNG_DVD-ROM_SD-608_____2.2_____\4&13B4AFD&0&0.0.0
   Name: SAMSUNG DVD-ROM SD-608
   Driver is running.
IDE\DISKWDC_WD204BA_____16.13M16\4457572D414D373032313633393820312
0202020
   Name: WDC WD204BA
   Driver is running.
ROOT\DMIO\0000
   Name: Logical Disk Manager
   Driver is running.
ROOT\FLOPPYDISK\0000
   Device has a problem: 28.
ROOT\FLOPPYDISK\0002
   Device has a problem: 01.
```

```
ROOT\FLOPPYDISK\0003
   Device has a problem: 01.
ROOT\FLOPPYDISK\0004
   Device is currently stopped.
ROOT\FTDISK\0000
   Name: Volume Manager
   Driver is running.
STORAGE\VOLUME\1&30A96598&0&SIGNATUREEA1AA9C7OFFSET1770DF800LENGTH3494AEA00
   Name: Generic volume
   Driver is running.
STORAGE\VOLUME\1&30A96598&0&SIGNATUREEA1AA9C7OFFSET7E00LENGTH1770CFC00
   Name: Generic volume
   Driver is running.
11 matching device(s) found on \\Server01.
```

Example 20: Find devices by hardware ID pattern

The following command uses the **DevCon Find** operation to search for mouse devices on Server01, a remote computer. Specifically, the command searches the Server01 computer for devices whose hardware ID or compatible ID includes "mou."

```
devcon /m:\\Server01 find *mou*
```

In this case, DevCon found both two mouse devices.

```
ROOT\*PNP0F03\1_0_21_0_31_0                 : Microsoft PS/2 Mouse
ROOT\RDP_MOU\0000                           : Terminal Server Mouse Driver
```

Because all DevCon display operations also find hardware IDs, you can use any display operation to search for hardware IDs. Select the operation based on the content that you need in the output. For example, to find the device drivers that mouse-related devices on a local computer use, submit the following command.

```
devcon driverfiles *mou*
```

In response, DevCon finds the devices and lists their drivers.

```
HID\VID_045E&PID_0039\6&DC36FDE&0&0000
   Name: Microsoft USB IntelliMouse Optical
   Driver installed from c:\windows\inf\msmouse.inf [HID_Mouse_Inst]. 2 file(s) used by d
river:
      C:\WINDOWS\System32\DRIVERS\mouhid.sys
      C:\WINDOWS\System32\DRIVERS\mouclass.sys
ROOT\RDP_MOU\0000
   Name: Terminal Server Mouse Driver
   Driver installed from c:\windows\inf\machine.inf [RDP_MOU]. 2 file(s) used by driver:
      C:\WINDOWS\System32\DRIVERS\termdd.sys
      C:\WINDOWS\System32\DRIVERS\mouclass.sys
2 matching device(s) found.
```

Example 21: Find devices by device instance ID or class

The following commands use the **DevCon Find** operation to display all legacy devices on the local computer. Because legacy devices do not have a hardware ID, you must search for them by their device instance ID (registry path), ROOT\LEGACY, or their setup class, LegacyDriver.

The first command finds legacy drivers by a device instance ID pattern. The ID pattern is prefaced by the at character (**@**) to indicate a device instance ID and followed by the wildcard character (*****) to find all devices in the ROOT\Legacy subkey.

```
devcon find @root\legacy*
```

The second command finds legacy devices by searching for all devices in the LegacyDriver class.

devcon find =legacydriver

Both commands produce the same output, in this case, finding the same 27 legacy devices.

```
ROOT\LEGACY_AFD\0000                    : AFD Networking Support Environment
ROOT\LEGACY_BEEP\0000                   : Beep
ROOT\LEGACY_DMBOOT\0000                    : dmboot
ROOT\LEGACY_DMLOAD\0000                    : dmload
ROOT\LEGACY_FIPS\0000                   : Fips
ROOT\LEGACY_GPC\0000                    : Generic Packet Classifier
ROOT\LEGACY_IPSEC\0000                  : ipsec
ROOT\LEGACY_KSECDD\0000                    : ksecdd
ROOT\LEGACY_MNMDD\0000                     : mnmdd
ROOT\LEGACY_MOUNTMGR\0000                     : mountmgr
ROOT\LEGACY_NDIS\0000                   : ndis
ROOT\LEGACY_NDISTAPI\0000                  : Remote Access NDIS TAPI Driver
ROOT\LEGACY_NDISUIO\0000                   : NDIS Usermode I/O Protocol
ROOT\LEGACY_NDPROXY\0000                   : NDProxy
ROOT\LEGACY_NETBT\0000                  : netbt
ROOT\LEGACY_NULL\0000                   : Null
ROOT\LEGACY_PARTMGR\0000                   : PartMgr
ROOT\LEGACY_PARVDM\0000                    : ParVdm
ROOT\LEGACY_RASACD\0000                    : Remote Access Auto Connection Driver
ROOT\LEGACY_RDPCDD\0000                    : RDPCDD
ROOT\LEGACY_RDPWD\0000                     : RDPWD
ROOT\LEGACY_TCPIP\0000                  : tcpip
ROOT\LEGACY_TDPIPE\0000                 : TDPIPE
ROOT\LEGACY_TDTCP\0000                  : TDTCP
ROOT\LEGACY_VGASAVE\0000                   : VgaSave
ROOT\LEGACY_VOLSNAP\0000                   : VolSnap
ROOT\LEGACY_WANARP\0000                    : Remote Access IP ARP Driver
27 matching device(s) found.
```

Example 22: Find (and find all) devices in a setup class
The following command uses the **DevCon FindAll** operation to find all devices on the computer in the Net setup class. The equal sign (**=**) indicates that Net is a setup class and not an ID.

devcon findall =net

In response, DevCon lists the following seven devices in the Net setup class. The first six are standard miniport driver devices. The seventh device, the RAS async adapter, is a software-enumerated device (SW*) that is not installed until it is needed.

```
PCI\VEN_10B7&DEV_9200&SUBSYS_00BE1028&REV_78\4&BB7B4AE&0&60F0: 3Com 3C920 Integrated Fast
Ethernet Controller (3C905C-TX Compatible)
ROOT\MS_L2TPMINIPORT\0000                  : WAN Miniport (L2TP)
ROOT\MS_NDISWANIP\0000                  : WAN Miniport (IP)
ROOT\MS_PPPOEMINIPORT\0000                  : WAN Miniport (PPPOE)
ROOT\MS_PPTPMINIPORT\0000                  : WAN Miniport (PPTP)
ROOT\MS_PTIMINIPORT\0000                : Direct Parallel
SW\{EEAB7790-C514-11D1-B42B-00805FC1270E}\ASYNCMAC      : RAS Async Adapter
7 matching device(s) found.
```

The following command compares the **DevCon Find** and **DevCon FindAll** operations by running a **DevCon Find** command with the same parameters as the previous **DevCon FindAll** command.

devcon find =net

In response, DevCon lists the following six devices in the Net setup class.

PCI\VEN_10B7&DEV_9200&SUBSYS_00BE1028&REV_78\4&BB7B4AE&0&60F0: 3Com 3C920 Integrated Fast Ethernet Controller (3C905C-TX Compatible)
ROOT\MS_L2TPMINIPORT\0000 : WAN Miniport (L2TP)
ROOT\MS_NDISWANIP\0000 : WAN Miniport (IP)
ROOT\MS_PPPOEMINIPORT\0000 : WAN Miniport (PPPOE)
ROOT\MS_PPTPMINIPORT\0000 : WAN Miniport (PPTP)
ROOT\MS_PTIMINIPORT\0000 : Direct Parallel
6 matching device(s) found.

Predictably, the **DevCon Find** command, which returns only currently installed devices, does not list the software-enumerated device because the device is not installed.

Example 23: Display the filter drivers for a setup class

The following command uses the **DevCon ClassFilter** operation to display the upper filter drivers for the DiskDrive setup class. Because this command includes no classfilter operators, DevCon displays the filter drivers for the class, but does not change them.

devcon classfilter DiskDrive upper

In response, DevCon displays the upper filter drivers for the DiskDrive class and confirms that it did not change them. In this case, the display shows that devices in the DiskDrive setup class use the PartMgr.sys upper filter driver.

Class filters unchanged.
PartMgr

Example 24: Add a filter driver to a setup class

The following command uses the **DevCon ClassFilter** operation to add a fictitious filter, Disklog.sys, to the list of upper filter drivers for the DiskDrive setup class.

This command uses the add-after (**+**) ClassFilter operator to load the Disklog driver after the PartMgr driver so that it receives data that PartMgr.sys has already processed.

When the command starts, the virtual cursor is positioned before the first filter driver. Because it is not positioned on a particular driver, DevCon adds the Disklog driver to the end of the filter driver list.

The command also uses the **/r** parameter, which reboots the system if it is necessary to make the class filter change effective.

devcon /r classfilter DiskDrive upper +Disklog

In response, DevCon displays the current upper filter drivers for the DiskDrive class.

Class filters changed. Class devices must be restarted for changes to take effect.
PartMgr
Disklog

If you misspell the driver name, or try to add a driver that isn't installed on the system, the command fails. DevCon does not add a driver unless the driver is registered as a service, that is, unless the driver has a subkey in the Services registry subkey (HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services).

The following command tests this safeguard feature. It attempts to add "Disklgg" (instead of "Disklog") to the list of upper filters for the DiskDrive class. The output demonstrates that the command fails.

devcon /r classfilter DiskDrive upper +Disklgg
devcon failed.

Example 25: Insert a filter driver in the class list

The following command uses the **DevCon ClassFilter** operation to add a fictitious filter driver, MyFilter.sys, to the list of upper filter drivers for the DiskDrive setup class. The command places MyFilter.sys between PartMgr.sys and Disklog.sys in the load order.

devcon /r classfilter DiskDrive upper @Disklog -MyFilter

The following list shows the filter drivers for the DiskDrive class before the command is submitted.

PartMgr
Disklog

The first subcommand, **@Disklog**, uses the positioning operator (**@**) to place the virtual cursor on the Disklog filter driver. The second subcommand, **-MyFilter**, uses the add-before operator (**-**) to add MyFilter.sys before Disklog.sys.
The command also uses the **/r** parameter, which reboots the system if it is necessary to make the class filter change effective.
The positioning operator is essential in this example. Before DevCon processes any classfilter subcommands, the virtual cursor is at the beginning of the list and is not positioned on any filter drivers. If you use the add-before (**+**) operator when the cursor is not on positioned on a driver, DevCon adds the driver to the beginning of the list. If you use the add-after (**-**) operator when the cursor is not positioned on a driver, it adds the driver to the end of the list.
In response, DevCon displays the current upper filter drivers for the DiskDrive class.

Class filters changed. Class devices must be restarted for changes to take effect.
PartMgr
MyFilter
Disklog

You can also use the following command to add the MyFilter driver and to place it between PartMgr and Disklog. In this example, the first subcommand, **@PartMgr**, positions the virtual cursor on the PartMgr filter driver. The second subcommand, **+MyFilter**, uses the add-after operator (+) to add MyFilter.sys after PartMgr.

devcon /r classfilter DiskDrive upper @PartMgr +MyFilter

Example 26: Replace a filter driver
The following command uses the **DevCon ClassFilter** operation to replace the original copy of MyFilter.sys with a new and improved version, MyNewFilter.sys, in the list of filter drivers for the DiskDrive setup class.

devcon /r classfilter DiskDrive upper !MyFilter +MyNewFilter

The following list shows the filter drivers for the DiskDrive class before the command is submitted.

PartMgr
MyFilter
Disklog

The first subcommand uses the delete operator (**!**) to delete MyFilter from the list of upper filter drivers for the DiskDrive class. (It does not affect the MyFilter.sys file in the C:\Windows\System32\Drivers directory.)
The second subcommand uses the add-after operator (**+**) to place the new filter driver in the position that the deleted driver occupied. Because the delete operator leaves the cursor in the position that the deleted filter occupied, the add-before (**-**) and add-after (**+**) operators have the same effect.)
The command also uses the **/r** parameter, which reboots the system if it is necessary to make the class filter change effective.
In response, DevCon shows the new class filter configuration for the DiskDrive class.

Class filters changed. Class devices must be restarted for changes to take effect.
PartMgr
MyNewFilter

Disklog

Example 27: Change the order of filter drivers
The following command uses the **DevCon ClassFilter** operation to change the order of filter drivers for the DiskDrive setup class. Specifically, it reverses the order of the second and third filter drivers.

```
devcon /r classfilter DiskDrive upper !Disklog =@PartMgr +Disklog
```

The following list shows the filter drivers for the DiskDrive class before the command is submitted. It also shows the intended result of the command.

Before	After
PartMgr	PartMgr
MyNewFilter	Disklog
Disklog	MyNewFilter

The first subcommand uses the delete operator (**!)** to delete Disklog from the list. The second subcommand uses the start operator (**=)** to move the virtual cursor back to the starting position and then uses the positioning operator (**@)** to place the cursor on the PartMgr driver. The start operator is necessary because the virtual cursor moves only forward through the list. The final subcommand uses the add-after operator (**+)** to add Disklog after PartMgr.
In response, DevCon shows the new class filter configuration for the DiskDrive class.

```
Class filters changed. Class devices must be restarted for changes to take effect.
  PartMgr
  Disklog
  MyNewFilter
```

Example 28: Enable a particular device
The following command uses the **DevCon Enable** operation to enable a programmable interrupt controller that had been disabled to correct a system problem. Because the controller hardware ID *PNP0000 includes an asterisk, the command uses the single quote character (') to direct DevCon to find the hardware ID precisely as it is specified in the command. Otherwise, the asterisk would be interpreted as a wildcard character.

```
devcon enable '*PNP0000
```

In response, DevCon displays the device instance ID of the device and explains that you must reboot the system to enable the device.

```
ACPI\PNP0000\4&B4063F4&0                : Enabled on reboot
Not all of 1 device(s) enabled, at least one requires reboot to complete the operation.
```

You can respond by rebooting the system, either manually, or by using the **DevCon Reboot** operation.
The following command adds the **/r** parameter to the previous command. The **/r** parameter reboots the system only if rebooting is required to complete an operation.

```
devcon /r enable '*PNP0000
```

In response, DevCon enables the device and then reboots the system to make the enabling effective.
When the system starts, use a DevCon status command to confirm that the device is enabled.

```
devcon status '*PNP0000
```

```
ACPI\PNP0000\4&B4063F4&0
  Name: Programmable interrupt controller
  Driver is running.
```

Example 29: Enable devices by class

The following command enables all printer devices on the computer by specifying the Printer setup class in a **DevCon Enable** command. The command includes the **/r** parameter, which reboots the system if it is necessary to make the enabling effective.

```
devcon /r enable =Printer
```

In response, DevCon displays the device instance ID of the printer that it found in the Printer class and reports that it is enabled. Although the command included the **/r** parameter, the system did not reboot because a reboot was not required to enable the printer.

```
LPTENUM\HEWLETT-PACKARDDESKJET_1120C\1&7530F08&0&LPT1.4       : Enabled
1 device(s) enabled.
```

Example 30: Disable devices by an ID pattern
The following command uses the **DevCon Disable** operation to disable the USB devices on the local computer. It identifies the devices by a hardware ID pattern (USB*). This pattern will match any device whose hardware ID or compatible ID begins with "USB." The command includes the **/r** parameter, which reboots the system if it is necessary to make the disabling effective.
Note Before using an ID pattern to disable a device, determine which devices will be affected. To do so, use the pattern in a display command, such as **devcon status USB*** or **devcon hwids USB***.

```
devcon /r disable USB*
```

In response, DevCon displays the device instance IDs of the USB devices and reports that they are disabled. Although the command included the **/r** parameter, the system did not reboot because a reboot was not required to disable the devices.

```
USB\ROOT_HUB\4&2A40B465&0
: Disabled
USB\ROOT_HUB\4&7EFA360&0
: Disabled
USB\VID_045E&PID_0039\5&29F428A4&0&2
: Disabled
3 device(s) disabled.
```

Example 31: Disable devices by device instance ID
The following command uses the **DevCon Disable** operation to disable the USB devices on the local computer. This command identifies the devices by their device instance IDs as indicated by the at character (**@**) that precedes each ID. Each device instance ID is separated from the others by a space.
Also, because the device instance IDs include the ampersand character (**&**), they are enclosed in quotation marks. The command includes the **/r** parameter, which reboots the system if it is necessary to make the disabling effective.

```
devcon /r disable "@USB\ROOT_HUB\4&2A40B465&0" "@USB\ROOT_HUB\4&7EFA360&0"
"@USB\VID_045E&PID_0039\5&29F428A4&0&2"
```

In response, DevCon displays the device instance IDs of the USB devices and reports that they are disabled. Although the command included the **/r** parameter, the system did not reboot because a reboot was not required to disable the devices.

```
USB\ROOT_HUB\4&2A40B465&0
: Disabled
USB\ROOT_HUB\4&7EFA360&0
: Disabled
USB\VID_045E&PID_0039\5&29F428A4&0&2
: Disabled
3 device(s) disabled.
```

Example 32: Update the driver for communication ports
The following command uses the **DevCon Update** operation to replace the current device driver for communication ports on the system with a test driver specified in the test.inf file. The command affects only devices whose entire hardware ID is *PNP0501 (including the asterisk).
You can use this command to replace signed drivers on the system with alternate drivers for testing or troubleshooting, or to associate the devices with the newest version of the same drivers.

devcon update c:\windows\inf\test.inf *PNP0501

In response, DevCon displays a **Hardware Installation** warning explaining that the driver has not passed Windows Logo testing. If you click the **Continue Anyway** button on the dialog box, the installation continues.
Then, DevCon displays the following success message.

Updating drivers for *PNP0501 from c:\windows\inf\test.inf.
Drivers updated successfully.

You can also use the **DevCon UpdateNI** operation, the noninteractive version of the **DevCon Update** operation, to update drivers. The **DevCon UpdateNI** operation is identical to the **DevCon Update** operation except that it suppresses all user prompts that require a response and assumes the default response to the prompt.
The following command uses the **DevCon UpdateNI** operation to install the test driver.

devcon updateni c:\windows\inf\test.inf *PNP0501

In this case, DevCon does not display the **Hardware Installation** warning. Instead, it assumes the default response, **Stop Installation**. As a result, DevCon cannot update the drivers and displays a failure message.

Updating drivers for *PNP0501 from c:\windows\inf\test.inf.
devcon failed.

Example 33: Install a device
The following command uses the **DevCon Install** operation to install a keyboard device on the local computer. The command includes the full path to the INF file for the device (keyboard.inf) and a hardware ID (*PNP030b).

devcon /r install c:\windows\inf\keyboard.inf *PNP030b

In response, DevCon reports that it has installed the device, that is, it has created a device node for the new device and updated the driver files for the device.

Device node created. Install is complete when drivers files are updated...
Updating drivers for *PNPO30b from c:\windows\inf\keyboard.inf
Drivers updated successfully.

Example 34: Install a device using unattended setup
The following example shows how to install the Microsoft Loopback Adapter during an unattended installation of Microsoft Windows XP.
To install this device during an unattended setup, begin by adding the following files to a floppy disk: devcon.exe and netloop.inf (C:\Windows\inf\netloop.inf).
Then, to the **[GUIRunOnce]** section of the unattended setup file, add the following DevCon command:

a:\devcon /r install a:\Netloop.inf '*MSLOOP

This command identifies the loopback adapter by using its hardware ID, *MSLOOP. The single quote character preceding "*MSLOOP" tells DevCon to interpret the string literally, that is, to interpret the asterisk as part of the hardware ID, not as a wildcard character.
The command also specifies that DevCon use the Netloop.inf file (on the floppy disk) in the installation. The **/r** parameter reboots the computer only if a reboot is necessary to complete the installation.
Finally, add network configuration settings to the unattended setup file and run the unattended setup.
Example 35: Remove devices by device instance ID pattern

The following command uses the **DevCon Remove** operation to remove all USB devices from the computer. It identifies the devices by a device instance ID pattern that matches any device instance ID (registry path) that begins with the "USB\" string. The at character (**@**) distinguishes the device instance ID from a hardware ID or compatible ID. The command also includes the **/r** parameter that reboots the system if it is required to make the remove procedure effective.
Warning Before removing any devices by using a pattern, determine which devices are affected. To do so, use the pattern in a display command, such as **devcon status @usb*** or **devcon hwids @usb***.

```
devcon /r remove @usb\*
```

In response, DevCon displays the device instance ID of the devices that it removed.

```
USB\ROOT_HUB\4&2A40B465&0                 : Removed
USB\ROOT_HUB\4&7EFA360&0                  : Removed
USB\VID_045E&PID_0039\5&29F428A4&0&2       : Removed
3 device(s) removed.
```

Example 36: Remove a particular network device
The following command uses the **DevCon Remove** operation to uninstall the NDISWAN miniport driver from the local computer. The command specifies the Net class and then refines the search by specifying devices in the class whose hardware ID or compatible ID include "ndiswan." The command also includes the **/r** parameter, which reboots the system if rebooting is required to make the remove procedure effective.
Warning Before removing any devices by using a pattern, determine which devices will be affected. To do so, use the pattern in a display command, such as **devcon status =net *ndiswan*** or **devcon hwids =net *ndiswan***.

```
devcon /r remove =net *ndiswan*
```

In response, DevCon displays the device instance ID of the device that it removed.

```
ROOT\MS_NDISWANIP\0000 : Removed 1 device(s) removed.
```

Example 37: Scan the computer for new devices
The following command use the **DevCon Rescan** operation to scan the local computer for new devices.

```
devcon rescan
```

In response, DevCon reports that it scanned the system but found no new devices.

```
Scanning for new hardware.
Scanning completed.
```

You can also use a **DevCon Rescan** command on a remote computer. The following command runs the **DevCon Rescan** operation on Server01, a remote computer, by adding the **/m** parameter to the command.

```
devcon /m:\\server01 rescan
```

Example 38: Restart a device
The following command uses the **DevCon Restart** operation to restart the loopback adapter on the local computer. The command limits the search to the Net setup class and, within that class, specifies the device instance ID of the loopback adapter, **ROOT*MSLOOP\0000**. The at character (**@**) identifies the string as an device instance ID. The single quote character ('), which requests a literal search, prevents DevCon from interpreting the asterisk in the ID as a wildcard character.

```
devcon restart =net @'ROOT\*MSLOOP\0000
```

In response, DevCon displays the device instance ID of the device and reports the result.

```
ROOT\*MSLOOP\0000                         : Restarted
```

1 device(s) restarted.

Example 39: Reboot the local computer
The following command uses the **DevCon Reboot** operation to reboot the operating system on the local computer and to associate the reboot with a hardware installation. Unlike the **/r** parameter, the **DevCon Reboot** operation does not depend on the return code from another operation.
You can include this command in scripts and batch files that require the system to reboot.

```
devcon reboot
```

In response, DevCon displays a message indicating that it is restarting the computer (Rebooting local machine).

DevCon uses the standard **ExitWindowsEx** function to reboot. If the user has open files on the computer or a program will not close, the system does not reboot until the user has responded to system prompts to close the files or end the process.

Example 40: Assign a hardware ID to a legacy device
The following command uses the **DevCon SetHwID** operation to assign the hardware ID, beep, to the legacy beep device.
The command uses the device instance ID of the device, ROOT\LEGACY_BEEP\0000, because the beep legacy device has no hardware IDs or compatible IDs. It uses the at character (**@**) to indicate that the string is a device instance ID.
The command does not use any symbol parameters to position the ID. By default, DevCon adds new hardware IDs to the end of a hardware ID list. In this case, because the device has no other hardware IDs, placement is irrelevant.

```
devcon sethwid @ROOT\LEGACY_BEEP\0000 := beep
```

In response, DevCon displays a message indicating that it is added beep to the hardware ID list for the device. It also displays the resulting hardware ID list. In this case, there is only one hardware ID in the list.

```
ROOT\LEGACY_BEEP\0000                : beep
Modified 1 hardware ID(s).
```

Example 41: Add a hardware ID to all legacy devices on a remote computer
The following command uses the **DevCon SetHwID** operation to add the hardware ID, legacy, to the list of hardware IDs for all legacy devices on the Server1 remote computer.
The command uses the **-** symbol parameter to add the new hardware ID to the end of the hardware ID list for the device, in case a preferred hardware ID has been created for one of the devices. It uses the **/m** parameter to specify the remote computer. It also uses a device instance ID pattern, **@ROOT\LEGACY***, to identify the legacy devices on the computer, that is, all devices whose device instance ID begins with **ROOT\LEGACY**.

```
devcon /m:\\Server1 sethwid @ROOT\LEGACY* := -legacy
```

In response, DevCon displays the resulting hardware ID lists for all affected devices.

```
ROOT\LEGACY_AFD\0000                  : legacy
ROOT\LEGACY_BEEP\0000                 : beep,legacy
ROOT\LEGACY_CRCDISK\0000              : legacy
ROOT\LEGACY_DMBOOT\0000               : legacy
ROOT\LEGACY_DMLOAD\0000               : legacy
ROOT\LEGACY_FIPS\0000                 : legacy
...
ROOT\LEGACY_WANARP\0000               : legacy
Modified 27 hardware ID(s).
```

After you assign the same hardware ID to a group of devices, you can use the other DevCon operations to view and change the devices in a single command.
For example, the following command displays the status of all legacy devices.

```
devcon status legacy
```

Example 42: Delete a hardware ID from all legacy devices on a remote computer
The following command uses the **DevCon SetHwID** operation to delete the hardware ID, **legacy**, from the list of hardware IDs for all legacy devices on the Server1 remote computer.
The command uses the **/m** parameter to specify the remote computer. It uses the hardware ID, **legacy**, to identify all devices that have that hardware ID. Then, it uses the **!** symbol parameter to delete the **legacy** hardware ID.

```
devcon /m:\\Server1 sethwid legacy := !legacy
```

In response, DevCon displays the resulting hardware ID lists for all affected devices.

```
ROOT\LEGACY_AFD\0000                    :
ROOT\LEGACY_BEEP\0000                   : beep
ROOT\LEGACY_CRCDISK\0000                :
ROOT\LEGACY_DMBOOT\0000                 :
ROOT\LEGACY_DMLOAD\0000                 :
ROOT\LEGACY_FIPS\0000                   :
...
ROOT\LEGACY_WANARP\0000                 :
Modified 27 hardware ID(s).
```

Example 43: Add, delete, and replace hardware IDs
The following series of examples shows how to use the varied features of the **DevCon SetHwID** operation.
This series uses a fictitious device, DeviceX, with the device instance ID, **ROOT\DeviceX\0000**. Before using DevCon, the device had the following list of hardware IDs:

```
Hw3 Hw4
```

The following command uses the **+** symbol to add **Hw1** and **Hw2** to the beginning of a list of hardware IDs for DeviceX. Because **Hw2** already appears in the list, it is moved, not added. The command identifies the device by its device instance ID, as indicated by the at character (**@**) preceding the ID.

```
devcon sethwid @ROOT\DEVICEX\0000 := +Hw1 Hw2
```

In response, DevCon displays the new hardware ID list for the device. Note that **Hw1** and **Hw2** appear at the beginning of the list in the specified order.

```
ROOT\DEVICEX\0000               : Hw1,Hw2,Hw3,Hw4
Modified 1 hardware ID(s).
```

Also, DevCon reports that it modified one hardware ID list, that is, the hardware ID list of one device.
The following command uses the **!** symbol to delete the **Hw1** hardware ID. It then lists the hardware ID, **Hw5**, without a symbol parameter. Without symbol parameters, SetHwID adds the hardware ID to the end of the hardware ID list for the device.
This command demonstrates that, unlike the other symbol parameters for the **DevCon SetHwID** operation, the **!** symbol applies only to the hardware ID that it prefixes.

```
devcon sethwid @ROOT\DeviceX\0000 := !Hw1 Hw5
```

In response, DevCon displays the resulting hardware ID list for DeviceX.

```
ROOT\DEVICEX\0000               : Hw2,Hw3,Hw4,Hw5
Modified 1 hardware ID(s).
```

The following command uses the = parameter to replace all hardware IDs in the list for DeviceX with a single hardware ID, **DevX**.

```
devcon sethwid @ROOT\DeviceX\0000 := =DevX
```

In response, DevCon displays the resulting hardware ID list for DeviceX.

```
ROOT\DEVICEX\0000                : DevX
Modified 1 hardware ID(s).
```

The success message indicates that DevCon modified the hardware ID of one device.

Example 44: Forcibly update the HAL

The following example shows how to use DevCon to update the HAL on the computer. In this example, a tester wants to replace the uniprocessor APCI APIC HAL that is best suited to the computer with a multiprocessor APCI APIC HAL for testing purposes.

The first command uses the **DevCon SetHwID** operation to change the hardware ID of the HAL from **acpiapic_up**, the hardware ID for uniprocessor HALs, to **acpiapic_mp**, the hardware ID for multiprocessor HALs.

You must change the hardware ID because the INF file for the HAL includes drivers for both uniprocessor and multiprocessor HALs. The system selects the most appropriate driver from the INF file based on the hardware ID of the device. If you do not change the hardware ID, then the **DevCon Update** command would simply reinstall the uniprocessor HAL driver.

In the following command, the command identifies the HAL by its instance ID, **ROOT\ACPI_HAL\0000**, as indicated by the **@** character preceding the ID. The command uses the **+** character to make **acpiapic_mp** the first hardware ID in the list for the HAL. Then, it uses the **!** character to delete the **acpiapic_up** hardware ID from the list of IDs for the HAL.

```
devcon sethwid @ROOT\ACPI_HAL\0000 := +acpiapic_mp !acpiapic_up
```

In response, DevCon displays the following new hardware ID list for the HAL.

```
ROOT\ACPI_HAL\0000               : acpiapic_mp
Modified 1 hardware ID(s).
```

The following command uses the **DevCon Update** operation to update the driver for the HAL.

```
devcon update c:\windows\inf\hal.inf acpiapic_mp
```

Then, DevCon displays the following success message.

```
Updating drivers for acpiapic_mp from c:\windows\inf\hal.inf.
Drivers updated successfully.
```