CHAPTER

W2

Dealing with Devices

The name of this chapter is "Dealing with Devices," but of course we never deal with our devices directly. Instead, we delegate that job to Windows, and it takes care of the behind-the-scenes dirty work of getting myriad devices to do we want them to do (or, at least, what they're supposed to do).

So we always seem to deal with our devices from a certain distance, sometimes frustratingly so. I hope the tweaks in this chapter will give you a greater sense of connection to your devices or, if that's too much to ask, that they'll at least help you out of a jam or two.

IN THIS CHAPTER

Reprogram a Key on Your Keyboard

When you press a key on your keyboard, the device generates a number that's unique to the key and transmits that number to Windows via the keyboard device driver. When Windows gets the number, it translates that value into the actual keypress and then performs whatever makes sense in the context (such as adding a letter to a text document).

This unique number that the keyboard generates for each key is called a *scancode*. Interestingly, you can tweak Windows so that it interprets a particular scancode in a different way.

For example, if your keyboard doesn't have a Windows Logo key, you can remap an existing key to act as a Windows Logo key. Similarly, if you never use Caps Lock, you can reprogram it to act as some other key (such as the lowercase A, which effectively eliminates long runs of uppercase letters thanks to an accidental press of Caps Lock). Finally, if you have a nonfunctioning key, you might also want to remap some little-used key to generate the character associated with the nonfunctioning key.

The trick in each case is to tell Windows to take the built-in scancode of an existing key and convert it to the scancode associated with another key. For example, suppose you want to reprogram the right Alt key to act like the Windows Logo key. When you press right Alt, the hexadecimal scancode E038 is generated. The scancode associated with the right Windows Logo key is E05C. Therefore, you need to tell Vista that whenever it detects the scancode E038 after a right Alt keypress, it should send to the system the code E05C, instead. This means that pressing the right Alt key will be the same thing as pressing the Windows Logo key.

To do this, open the Registry Editor and navigate to the following key:

```
HKLM\SYSTEM\CurrentControlSet\Control\Keyboard Layout
```

Select, Edit, New, Binary Value, type **`Scancode Map`**, and press Enter. The `Scancode Map` setting uses the following structure:

Section	Bytes	Example
Version	4	00 00 00 00
Flags	4	00 00 00 00
Total Mappings	4	02 00 00 00 (2 mappings)
Mappings	4	5C E0 38 E0 (right Alt mapped to right Windows Logo)
Terminator	4	00 00 00 00

Note that some of these values are a bit odd because they reverse the order of the bytes. For example, you enter the hexadecimal value E05C as 5C E0.

If you're remapping just one key, set the total mappings to 2 (including the null terminator). For example, the following `Scancode Map` value remaps right Alt to right Windows Logo:

```
00 00 00 00 00 00 00 00
02 00 00 00 5C E0 38 E0
00 00 00 00
```

To add more mappings, adjust the total mappings value and add the mappings between the Total Mappings section and the Terminator section. For example, the following `Scancode Map` value remaps right Alt to right Windows Logo *and* Caps Lock (scancode 3A) to the letter A (scancode 1E):

```
00 00 00 00 00 00 00 00
03 00 00 00 5C E0 38 E0
1E 00 3A 00 00 00 00 00
```

Figure W2.1 shows how this looks in the Edit Binary Value dialog box. You need to reboot your computer to put the new key mapping into effect.

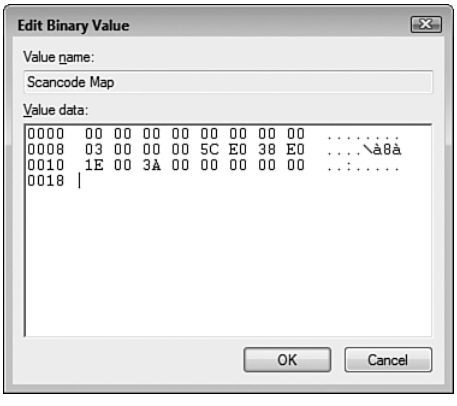

FIGURE W2.1

This `Scancode Map` *value remaps right Alt to right Windows Logo and Caps Lock to lowercase A.*

Table W2.1 lists the standard keyboard scancodes.

Table W2.1 Standard Keyboard Scancodes

Key	Scancode
Alphanumeric keys	
1	02
2	03
3	04
4	05
5	06
6	07
7	08
8	09
9	0A
0	0B
a	1E
b	30
c	2E
d	20
e	12
f	21
g	22
h	23
i	17
j	24
k	25
l	26
m	32
n	31
o	18
p	19

Key	Scancode
q	10
r	13
s	1F
t	14
u	16
v	2F
w	11
x	2D
y	15
z	2C
`	29
-	0C
=	0D
Backspace	0E
Tab	0F
[	1A
]	1B
\	2B
;	27
‘	28
#	2B
Enter	1C
\	56
,	33
.	34
/	35
Space Bar	39
Function Keys	
F1	3B
F2	3C

continues

Table W2.1 Continued

Key	Scancode
F3	3D
F4	3E
F5	3F
F6	40
F7	41
F8	42
F9	43
F10	44
F11	57
F12	58
Modifier Keys	
Caps Lock	3A
Left Shift	2A
Right Shift	36
Left Ctrl	1D
Left Alt	38
Right Alt	E0 38
Right Ctrl	E0 1D
Num Lock	45
Scroll Lock	46
Esc	01
Print Screen	E0 2A E0 37
Pause	E1 1D 45 E1 9D C5
Navigation Keys	
Insert	E0 52
Delete	E0 53
Home	E0 47
End	E0 4F
Page Up	E0 49

Key	Scancode
Page Down	E0 51
Up Arrow	E0 48
Down Arrow	E0 50
Left Arrow	E0 4B
Right Arrow	E0 4D
Numeric Keypad Keys	
Home	47
End	4F
Page Up	49
Page Down	51
Insert	52
Delete	53
Up Arrow	48
Down Arrow	50
Left Arrow	4B
Right Arrow	4D
/	E0 35
*	37
-	4A
+	4E
Enter	E0 1C
Windows Keys	
Left Windows	E0 5B
Right Windows	E0 5C
Context Menu	E0 5D

Display a List of Nonworking Devices

Device Manager not only provides you with a comprehensive summary of your system's hardware data, it also doubles as a decent troubleshooting tool. That's because Device Manager uses three icons to give you an indication of the device's current status:

- A black exclamation mark (!) on a yellow field tells you that there's a problem with the device.
- A red X tells you that the device is disabled or missing.
- A blue i on a white field tells you that the device's Use Automatic Settings check box (on the Resources tab) is deactivated and that at least one of the device's resources was selected manually. Note that the device might be working just fine, so this icon doesn't indicate a problem. If the device isn't working properly, however, the manual setting might be the cause. (For example, the device might have a DIP switch or jumper set to a different resource.)

That's great, but it's not always convenient to fire up Device Manager to check for problems. To avoid that, use the script in Listing W2.1, which displays a list of all the problem devices on your system.

> **NOTE** The file containing the script in Listing W2.1—`ListNonWorkingDevices.vbs`—is available from my website at http://mcfedries.com/cs/content/TweakItFreakIt.aspx. See Chapter 37, "Running Scripts," to learn how to run the script on your PC.

Listing W2.1 A Script That Displays a List of a PC's Nonworking Devices

```
Option Explicit
Dim strComputer, objWMI, collDevices, objDevice
Dim intDevices, strMessage
'
' Get the WMI object
'
strComputer = "."
Set objWMI = GetObject("winmgmts:\\" & _
             strComputer & "\root\cimv2")
'
' Return the collection of nonworking devices on the computer
'
Set collDevices = objWMI.ExecQuery _
```

```
    ("Select * from Win32_PnPEntity " _
        & "WHERE ConfigManagerErrorCode <> 0")
'
' Run through each item in the collection
'
intDevices = 0
strMessage = ""
For Each objDevice in collDevices
    strMessage = strMessage & "Device Name: " & objDevice.Name & vbCrLf
    strMessage = strMessage & "Manufacturer: " &
      ➥objDevice.Manufacturer & vbCrLf
    strMessage = strMessage & "Device ID: " & objDevice.DeviceID &
      ➥vbCrLf
    strMessage = strMessage & "Service: " & objDevice.Service & vbCrLf
    strMessage = strMessage & "Error Code: " &
      ➥objDevice.ConfigManagerErrorCode & vbCrLf
    strMessage = strMessage & vbCrLf & vbCrLf
    intDevices = intDevices + 1
Next
'
' Display the results
'
If intDevices = 0 Then
    WScript.Echo "No non-working devices found!"
Else
    WScript.Echo "Found " & intDevices & " non-working device(s):" & _
                 vbCrLf & vbCrLf & _
                 strMessage
End If
```

The script sets up the Windows Management Instrumentation (WMI) object and then uses WMI to return the collection of nonworking devices (that is, where the device's `ConfigManagerErrorCode` property isn't 0). A `For Each...Next` loop goes through each device and stores various data about the device to a string variable, including the device name and ID, and the error code. The script then displays the results, and Figure W2.2 shows an example.

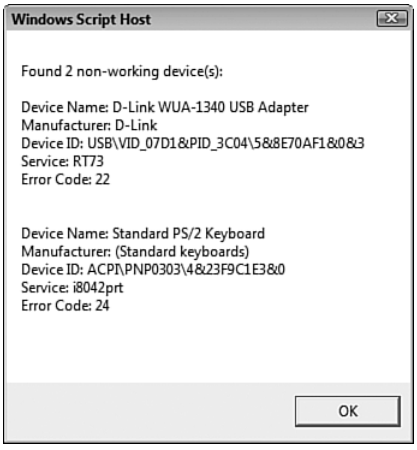

FIGURE W2.2

Sample output from the script in Listing W2.1.

Table W2.2 lists the various error codes and what they mean.

Table W2.2 Error Codes for Nonworking Devices

Code	Description
0	Device is working properly.
1	Device is not configured correctly.
2	Windows cannot load the driver for this device.
3	Driver for this device might be corrupted, or the system may be low on memory or other resources.
4	Device is not working properly. One of its drivers or the Registry might be corrupted.
5	Driver for the device requires a resource that Windows cannot manage.
6	Boot configuration for the device conflicts with other devices.
7	Cannot filter.
8	Driver loader for the device is missing.
9	Device is not working properly. The controlling firmware is incorrectly reporting the resources for the device.
10	Device cannot start.
11	Device failed.
12	Device cannot find enough free resources to use.
13	Windows cannot verify the device's resources.
14	Device cannot work properly until the computer is restarted.
15	Device is not working properly because of a possible reenumeration problem.

Code	Description
16	Windows cannot identify all the resources that the device uses.
17	Device is requesting an unknown resource type.
18	Device drivers must be reinstalled.
19	Failure using the Vloader.
20	Registry might be corrupted.
21	System failure. If changing the device driver is ineffective, see the hardware documentation. Windows is removing the device.
22	Device is disabled.
23	System failure. If changing the device driver is ineffective, see the hardware documentation.
24	Device is not present, not working properly, or does not have all its drivers installed.
25	Windows is still setting up the device, but the installation is incomplete.
26	Windows is still setting up the device, but not all the devices drivers were installed or there's a problem with one of the device drivers.
27	Device does not have valid log configuration.
28	Device drivers are not installed.
29	Device is disabled. The device firmware did not provide the required resources.
30	Device is using an IRQ resource that another device is using.
31	Device is not working properly. Windows cannot load the required device drivers.

Show Nonpresent Devices in Device Manager

Vista XP

When you open Device Manager (in Vista, select Start, type **`device`**, press Enter, and enter your UAC credentials; in XP, select Start, Run, type **`devmgmt.msc`**, and click OK), the list of devices you see includes only Plug and Play devices. If you have any non-Plug and Play devices that you want to work with, select View, Show Hidden Devices.

That works, but it doesn't mean that Device Manager is now showing all your devices. If you have any devices that you've installed in Windows, but that you regularly connect and then disconnect (such as a USB digital camera), Device Manager won't show them. (Windows describes such devices as *ghosted* devices.) That makes a bit of sense, because it might be confusing to see non-connected hardware in Device Manager.

However, what if you're having a problem with a ghosted device? For example, suppose Windows hangs or crashes every time you connect such a device.

Ideally you'd like to use Device Manager to uninstall that device, but you can't because Windows goes belly-up whenever you connect the nasty thing. What do to?

The solution to this kind of problem is to force Device Manager to show ghosted devices. Here's how:

1. In Vista, select Start, type **command**, and then click Command Prompt; in XP, select Start, Run to open the Run dialog box, type **cmd**, and then click OK. Windows launches a new Command Prompt session.
2. Type the following command and then press Enter:

   ```
   set devmgr_show_nonpresent_devices=1
   ```

3. Type the following command and then press Enter to launch Device Manager:

   ```
   devmgmt.msc
   ```

4. In Device Manager, select View, Show Hidden Devices. Device Manager adds to the device list any ghosted devices that are installed on your PC.

NOTE By setting the `DEVMGR_SHOW_NONPRESENT_DEVICES` environment variable in your Command Prompt session, you must launch Device Manager from that session. If you just launch Device Manager in the usual way, you won't see the ghosted devices.