PCIe Device Listing on Windows

With PCITree.ps1, the PCIe hierarchy is retrieved and represented either as console *highlighted* or *html*. IT personnel can use or expand the tool in support cases.

This excerpt represents the bulk:

```
$filter = "ConfigManagerErrorCode != $($script:ProblemNames.CM_PROB_PHANTOM) AND " +
          "(DeviceID LIKE ""PCI\\%"" OR DeviceID LIKE ""ACPI\\PNP0A08\\%"")";
$ap = Get-WmiObject Win32_PnPEntity -Filter $filter;
```

- *Phantom devices* are devnodes stored in registry without a physical adapter plugged in the system. These are skipped to prevent false positives.
- `ACPI\PNP0A08` root complexes are enumerated on UEFI systems. Legacy `ACPI\PNP0A03` are not represented.

For each device, be it root complex, PCIe switch or endpoint, a number of properties are displayed. The console options `-AsVT` or `-AsText` do not show the *driver stack*.

The script requires powershell 5.0 *Desktop* edition. `GetDeviceProperties` method is not available under *pwsh.exe Core* with `Get-CimInstance`. At the expense of a performance penalty, it can be replaced with `Get-PnPDeviceProperty` cmdlet and have full support for *Core*.

```
$ret = $ap | Select-Object `
    @{ Name="BARs";
    Expression={ $id = $_.DeviceID; ($ba | Where-Object { $_.DeviceID -eq $id }).BAR }
    },
    @{ Name="Parent";
    Expression={ $_.GetDeviceProperties("DEVPKEY_Device_Parent").deviceProperties.Data }
    };
```

An element in the hierarchy has one `DEVPKEY_Device_Parent`, multiple `Descendants`. Before computing the descendants, the list is sorted by BDF, then ACPI root complexes are given priority:

- *BDF* sort can place the RC at random indexes among PCIe devices with same `0:0.0` location.

On systems with multiple root complexes, the device ID has a hexadecimal suffix next to `ACPI\PNP0A08\`. Sorting by suffix keeps the tree representation consistent.

```
$List.Value = $List.Value | Sort-Object BDF;
$acpi = @($List.Value | Where-Object { $_.DeviceID -like "ACPI\PNP0A08\*" } | Sort-Object `
            @{ Expression={ $suffix = ($_.DeviceID -split "\\")[-1];
                            [Int]("0x$suffix")
                }
            });
$pci = $List.Value | Where-Object { $_.DeviceID -like "PCI\*" };
$List.Value = $acpi + $pci;
```

Base address registers are computed with `CM_Get_First_Log_Conf` and `CM_Get_Res_Des_Data` Win32APIs.

- `Win32_PnPAllocatedResource, Win32_DeviceMemoryAddress` associators lead to noise: the BARs are not unique, 64-bit BARs are truncated to 32-bit.

For brevity, `MEM_RESOURCE` structure is marked as *unsafe*: `MD_Alloc_Base`, `MD_Alloc_End` are padded.

```
$co = [System.CodeDom.Compiler.CompilerParameters]::new();
$co.CompilerOptions += "/unsafe";

Add-Type -CompilerParameters $co @"
    [StructLayout(LayoutKind.Sequential)]
    unsafe public struct MEM_RESOURCE
    {
        public UInt32 MD_Count;
        public UInt32 MD_Type;
        public UInt64 MD_Alloc_Base;
        public UInt64 MD_Alloc_End;
        public fixed UInt32 Unused[11];
    };
```

```
        [DllImport("cfgmgr32.dll")]
        public static extern UInt32
            CM_Get_Res_Des_Data(IntPtr ResDes,
                                ref MEM_RESOURCE Buffer,
                                UInt32 BufferLen,
                                UInt32 Flags);
"@;
```

`-AsHTML` cli switch is fully fledged: driver stack, NUMA node, problem code linked to documentation, number of processor packages are among the properties being displayed. *"Native hot-plug interrupts granted by firmware"* indicates platform support for adapter hot remove/add.

```
ImportNative;

$devs = GetPCIeDevNodes;
PCITree ([ref]$devs);

if ($PSCmdlet.ParameterSetName -eq "HTML") {
    $ct = RenderHTML $devs;
    GenerateFileName ([ref]$ct);
} else {
    PrintHeader;
    DisplayConsole $devs;
}
```

Notes

- Use `Set-ExecutionPolicy Bypass -Scope Process` before launching the script.
- Heavy `<table>` usage leads to gaps on rendering the contracted descendants.
- lspci windows is currently blacklisted by the browser.
- Large PCIe hierarchy with hundreds of devices takes 20+ seconds to be shown. A progress bar yields the devices enumerated until completion.
- `-AsVT` output can have its information stream redirected to a file. Coloring is preserved.

```
Invoke-Command (Get-PSSession) {
    .\PCITree.ps1 -AsVT 6>C:\results.txt;
}
Copy-Item -FromSession (Get-PSSession) C:\results.txt
Get-Content results.txt;
```