PowerShell: Utiliser les informations de WinSAT

Définition: WinSAT

Depuis l'arrivée de Microsoft Windows Vista, Microsoft sait s'adapter aux différents composants et exploiter certaines fonctionnalités matérielles. WinSAT peut être aussi utilisé pour savoir si la carte graphique est compatible avec l'interface Aero et donc l'activer. L'outil n'est pas exécuté automatiquement lors d'un changement de matériel. Il est accessible dans les panneaux de configuration où il fournit une note de 1 à 5.9 à votre PC (la valeur maximale pourra être mise à jour si les périphériques dépassent de trop cette valeur). Les jeux sous Vista peuvent profiter de ce système et préciser la note minimum et la note recommandée pour pouvoir jouer. Ce n'est cependant qu'à titre d'informations et rien n'empêche de jouer si les prérequis ne sont pas présents. Chaque grade correspond selon Microsoft à un type de recommandation matérielle. Certaines recommandations semblent contradictoires, c'est pourquoi le détail ne figure pas dans la liste ci-dessous.

Grade 1 : Ce grade correspond au minimum matériel requis pour l'exécution du système d'exploitation. Un ordinateur de ce type ne pourra être destiné qu'à des applications simples comme la bureautique ou la navigation sur Internet.

Grade 2 : Ce deuxième grade correspond à un ordinateur permettant d'effectuer les mêmes tâches qu'un ordinateur de Grade 1 mais dans des conditions, plus agréables. Ce second grade est identique au Grade 1 auquel s'ajouterait du multimédia sans pour autant atteindre la haute-définition. À noter que la non-conformité de la carte graphique DirectX 9 ne permettra pas à ce type de machine d'exécuter Aero dans de bonnes conditions. On peut toutefois, et malgré les recommandations de l'équipe de développement, forcer son exécution. Cela engendrera une baisse significative des performances, la grande majorité des effets appliqués n'étant pas gérés par le GPU.

Grade 3 : Voici l'ordinateur qui sert de base à la réception du logo Argent pour Vista. Il s'agit du modèle standard d'exécution de l'environnement du système d'exploitation. Ces ordinateurs pourront faire fonctionner la majorité des applications actuelles, y compris en haute-définition, suivant la résolution désirée et les caractéristiques précises de l'ordinateur.

Grade 4 : Ce grade désigne les ordinateurs sur lesquels le système d'exploitation s'exécutera dans de très bonnes conditions. Cette configuration correspond approximativement aux ordinateurs neufs d'entrée de gamme en avril 2006. Ces ordinateurs exécuteront Aero Glass correctement avec activation de la composition de l'affichage. Ils pourront faire fonctionner dans de bonnes conditions les éléments de Windows Media Center dans le cas d'une configuration à 2 tuners au plus. Le streaming haute-définition ne sera pas non plus géré. Les éléments déterminants véritablement de cette configuration restent la quantité de mémoire vive et la puissance de la carte graphique.

Grade 5 : Sur ces configurations, Le système d'exploitation s'exécutera avec le maximum d'effets et au maximum de ses possibilités. Ce type d'ordinateurs correspond au haut de gamme en juin 2006. Les ordinateurs qui sortiront par la suite et qui seront encore plus puissants resteront à ce grade de 5, reflétant globalement les possibilités intrinsèques de la machine.

WinSAT: Lancement initial

Si cette étape était obligatoire sous Windows 7, même si l'on pouvait la passer via un fichier réponse. Depuis il faudra passer cette commande sous Windows pour lancer cette :

```
winsat formal>>c:\windows\temp\WinSat.log
```

Quand la commande sera terminée, on pourra trouver les résultats dans le dossier suivant: “C:\Windows\Performance\WinSAT\DataStore”.

WinSAT: Exploitation avec PowerShell

Récupérer simplement les résultats

```
get-wmiobject Win32_WinSAT | SELECT-OBJECT CPUScore, MemoryScore,
GraphicsScore, D3DScore, DiskScore, WinSPRLevel
```

Récupérer les résultats d'un poste distant

```
Get-WmiObject -ComputerName 127.0.0.1 win32_winsat  | SELECT-OBJECT
CPUScore, MemoryScore, GraphicsScore, D3DScore, DiskScore, WinSPRLevel
```

Maintenant via une interface graphique

```
[xml]$gl_szXaml = @"
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Windows Experience Index (WEI) Scores"
    Width="333" Height="451"
    Background="lightgray"
    WindowStartupLocation="CenterScreen" Topmost="True"
ResizeMode="NoResize" ShowInTaskbar="True">
    <StackPanel>
        <Grid Margin="3">
            <Grid.ColumnDefinitions>
                <ColumnDefinition />
                <ColumnDefinition Width="Auto" />
                <ColumnDefinition Width="Auto" />
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
```

```
            </Grid.RowDefinitions>

            <TextBlock Grid.Column="0" Grid.Row="1" Text="CPU:"
FontWeight="Bold" Margin="3" />
            <TextBlock Grid.Column="0" Grid.Row="2" Text="RAM:"
FontWeight="Bold" Margin="3" />
            <TextBlock Grid.Column="0" Grid.Row="3" Text="Graphics:"
FontWeight="Bold" Margin="3" />
            <TextBlock Grid.Column="0" Grid.Row="4" Text="Gaming graphics:"
FontWeight="Bold" Margin="3" />
            <TextBlock Grid.Column="0" Grid.Row="5" Text="Primary hard
disk:" FontWeight="Bold" Margin="3" />
            <TextBlock Grid.Column="0" Grid.Row="6" Text="Global Indicator:"
FontWeight="Bold" Margin="3" />

            <TextBlock Grid.Column="1" Grid.Row="0" Text="Initial"
FontWeight="Bold" Margin="3" />
            <TextBox x:Name="tbCpuScoreCurr" Grid.Column="1" Grid.Row="1"
Text="{Binding CPUSCORE}" IsReadOnly="True" Margin="3" />
            <TextBox x:Name="tbRamScoreCurr" Grid.Column="1" Grid.Row="2"
IsReadOnly="True" Margin="3" />
            <TextBox x:Name="tbGraphScoreCurr" Grid.Column="1" Grid.Row="3"
IsReadOnly="True" Margin="3" />
            <TextBox x:Name="tbGraphGamingScoreCurr" Grid.Column="1"
Grid.Row="4" IsReadOnly="True" Margin="3" />
            <TextBox x:Name="tbHddScoreCurr" Grid.Column="1" Grid.Row="5"
IsReadOnly="True" Margin="3" />
            <TextBox x:Name="tbWinSPRLevelScoreCurr" Grid.Column="1"
Grid.Row="6" IsReadOnly="True" Margin="3" />

            <TextBlock Grid.Column="2" Grid.Row="0" Text="Recent"
FontWeight="Bold" Margin="3" />
            <TextBox x:Name="tbCpuScoreNew" Grid.Column="2" Grid.Row="1"
IsReadOnly="True" Margin="3" />
            <TextBox x:Name="tbRamScoreNew" Grid.Column="2" Grid.Row="2"
IsReadOnly="True" Margin="3" />
            <TextBox x:Name="tbGraphScoreNew" Grid.Column="2" Grid.Row="3"
IsReadOnly="True" Margin="3" />
            <TextBox x:Name="tbGraphGamingScoreNew" Grid.Column="2"
Grid.Row="4" IsReadOnly="True" Margin="3" />
            <TextBox x:Name="tbHddScoreNew" Grid.Column="2" Grid.Row="5"
IsReadOnly="True" Margin="3" />
            <TextBox x:Name="tbWinSPRLevelScoreNew" Grid.Column="2"
Grid.Row="6" IsReadOnly="True" Margin="3" />
        </Grid>

        <TextBox x:Name="tbParms" />

        <Button x:Name="bttnRun" Content="Re-Run" />
        <ListBox x:Name="lbStatus" MinHeight="200" MaxHeight="200" />
    </StackPanel>
```

```
</Window>
"@


#####
## Funcs
#####

function Get-WinSAT-Report-Latest-Name () {
    $oFile = Get-ChildItem $gl_szWinSATReportPath | Sort-Object
LastWriteTime -descending | Select-Object Name |
             Select-String -pattern "Formal.Assessment" | Select-Object -
First 1

    return $oFile.Line
}

function Wait-For-WinSAT-Report () {
    $szOldName = Get-WinSAT-Report-Latest-Name

    do {
        Start-Sleep -s 10
        $szNewName = Get-WinSAT-Report-Latest-Name
    }
    while ($szOldName -eq $szNewName)
}

function Set-Gui-Scores-Curr () {
    $tbCpuScoreCurr = $gl_oWnd.FindName("tbCpuScoreCurr")
    $tbRamScoreCurr = $gl_oWnd.FindName("tbRamScoreCurr")
    $tbGraphScoreCurr = $gl_oWnd.FindName("tbGraphScoreCurr")
    $tbGraphGamingScoreCurr = $gl_oWnd.FindName("tbGraphGamingScoreCurr")
    $tbHddScoreCurr = $gl_oWnd.FindName("tbHddScoreCurr")
    $tbWinSPRLevelScoreCurr = $gl_oWnd.FindName("tbWinSPRLevelScoreCurr")

    $oScoresCurr = Get-WmiObject Win32_WinSat | SELECT-OBJECT CPUScore,
MemoryScore, GraphicsScore, D3DScore, DiskScore, WinSPRLevel

    $tbCpuScoreCurr.DataContext = $oScoresCurr
    $tbRamScoreCurr.Text = $oScoresCurr.MEMORYSCORE
    $tbGraphScoreCurr.Text = $oScoresCurr.GRAPHICSSCORE
    $tbGraphGamingScoreCurr.Text = $oScoresCurr.D3DSCORE
    $tbHddScoreCurr.Text = $oScoresCurr.DISKSCORE
    $tbWinSPRLevelScoreCurr.Text = $oScoresCurr.WinSPRLevel
}

function Set-Gui-Scores-New () {
    $tbCpuScoreNew = $gl_oWnd.FindName("tbCpuScoreNew")
    $tbRamScoreNew = $gl_oWnd.FindName("tbRamScoreNew")
    $tbGraphScoreNew = $gl_oWnd.FindName("tbGraphScoreNew")
    $tbGraphGamingScoreNew = $gl_oWnd.FindName("tbGraphGamingScoreNew")
```

```
    $tbHddScoreNew = $gl_oWnd.FindName("tbHddScoreNew")
    $tbWinSPRLevelScoreNew = $gl_oWnd.FindName("tbHddScoreNew")

    $oScoresNew = Get-WmiObject Win32_WinSat | SELECT-OBJECT CPUScore,
MemoryScore, GraphicsScore, D3DScore, DiskScore, WinSPRLevel

    $tbCpuScoreNew.Text = $oScoresNew.CPUSCORE
    $tbRamScoreNew.Text = $oScoresNew.MEMORYSCORE
    $tbGraphScoreNew.Text = $oScoresNew.GRAPHICSSCORE
    $tbGraphGamingScoreNew.Text = $oScoresNew.D3DSCORE
    $tbHddScoreNew.Text = $oScoresNew.DISKSCORE
    $tbWinSPRLevelScoreNew.Text = $oScoresNew.WinSPRLevel

    $szMsg = "CPU: {0}, RAM: {1}, GRF: {2}, Gaming GRF: {3}, HDD: {4},
SCORE: {5}" -f $oScoresNew.CPUSCORE, $oScoresNew.MEMORYSCORE,
$oScoresNew.GRAPHICSSCORE, $oScoresNew.D3DSCORE, $oScoresNew.DISKSCORE,
$oScoresNew.WinSPRLevel
    Set-Gui-Status ($szMsg)
}

function Set-Gui-Status ($szMsg) {
    $lbStatus = $gl_oWnd.FindName("lbStatus")

    $lbStatus.Items.Add($szMsg)
}


#####
## Main
#####

$gl_szSysRoot = [environment]::GetEnvironmentVariable("systemroot")
$gl_szWinSATReportPath = "$gl_szSysRoot\Performance\WinSAT\DataStore"

Add-Type -As PresentationFramework
$gl_oReader=(New-Object System.Xml.XmlNodeReader $gl_szXaml)
$gl_oWnd=[Windows.Markup.XamlReader]::Load($gl_oReader)

Set-Gui-Scores-Curr

$gl_bttnRun = $gl_oWnd.FindName("bttnRun")
$gl_bttnRun.Add_Click({
    $oStartTS = Get-Date
    Set-Gui-Status ("Test started: " + $oStartTS)

    $gl_oWnd.Topmost = $false

    $szCmd = "winsat formal "+ $gl_oWnd.FindName("tbParms").Text
    Set-Gui-Status ("cmd: "+ $szCmd)
    Invoke-Expression ($szCmd)
```

```
    Wait-For-WinSAT-Report

    $gl_oWnd.Topmost = $true

    Set-Gui-Scores-New

    $oEndTS = Get-Date
    Set-Gui-Status ("Test completed: " + $oEndTS)
    $oElapsedTime = $oEndTS - $oStartTS
    Set-Gui-Status ("Elapsed time: {0}:{1}:{2}.{3}" -f  $oElapsedTime.Hours,
$oElapsedTime.Minutes, $oElapsedTime.Seconds, $oElapsedTime.Milliseconds)
    Set-Gui-Status ("")
})

$gl_oWnd.ShowDialog() | Out-Null
```