Récupérer les logiciels installés avec Powershell sur Windows

Via les clefs de registre

```
# Récupérer la liste via les clefs de registre
Get-ChildItem -Path
HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\ | Get-
ItemProperty |
Select-Object DisplayName, DisplayVersion
# Récupérer la liste via les clefs de registre Wow6432node avec une
vérification
$RegWow6432node =
'HKLM:\SOFTWARE\Wow6432node\Microsoft\Windows\CurrentVersion\Uninstall'
if((((Get-Item -Path $RegWow6432node).GetValue($value)) -ne $null) -eq
$false)
{
Get-ChildItem -Path $RegWow6432node | Get-ItemProperty |
Select-Object DisplayName, DisplayVersion | Format-List
}
```

Chocolatey

Si l'on utilise l'application Chocolatey, il est possible de récupérer la liste des applications installées avec lui.

```
if($env:Path -like '*chocolatey*') {
    choco list -lo
}
```

Eviter Get-CimInstance et Get-WMIObject

Utiliser la méthode Cim ou WMI, n'est pas conseillé car elle est très longue

WMI

```

```

```
Get-WmiObject -Class 'Win32_product'
```

CIM

```
Get-CimInstance -ClassName 'Win32_product'
```

Fonction pour récupérer à distance

```
function Get-InstalledSoftware
{
    <#
    .SYNOPSIS
        Retrieves a list of all software installed on a Windows computer.
    .EXAMPLE
        PS> Get-InstalledSoftware
        This example retrieves all software installed on the local computer.
    .PARAMETER ComputerName
        If querying a remote computer, use the computer name here.
    .PARAMETER Name
        The software title you'd like to limit the query to.
    .PARAMETER Guid
        The software GUID you'e like to limit the query to
    #>
    [CmdletBinding()]
    param (
        [Parameter()]
        [ValidateNotNullOrEmpty()]
        [string]$ComputerName = $env:COMPUTERNAME,
        [Parameter()]
        [ValidateNotNullOrEmpty()]
        [string]$Name,
        [Parameter()]
        [ValidatePattern('\b[A-F0-9]{8}(?:-[A-F0-9]{4}){3}-[A-F0-9]{12}\b')]
        [string]$Guid
    )
    process
    {
        try
        {
            $scriptBlock = {
                $args[0].GetEnumerator() | ForEach-Object { New-Variable -
Name $_.Key -Value $_.Value }
                $UninstallKeys =
"HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall",
```

```
"HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall"
                New-PSDrive -Name HKU -PSProvider Registry -Root
Registry::HKEY_USERS | Out-Null
                $UninstallKeys += Get-ChildItem HKU: | where { $_.Name -
match 'S-\d-\d+-(\d+-){1,14}\d+$' } | foreach {
"HKU:\$($_.PSChildName)\Software\Microsoft\Windows\CurrentVersion\Uninstall"
}
                if (-not $UninstallKeys)
                {
                    Write-Warning -Message 'No software registry keys found'
                }
                else
                {
                    foreach ($UninstallKey in $UninstallKeys)
                    {
                        $friendlyNames = @{
                            'DisplayName' = 'Name'
                            'DisplayVersion' = 'Version'
                        }
                        Write-Verbose -Message "Checking uninstall key
[$($UninstallKey)]"
                        if ($Name)
                        {
                            $WhereBlock = { $_.GetValue('DisplayName') -like
"$Name*" }
                        }
                        elseif ($GUID)
                        {
                            $WhereBlock = { $_.PsChildName -eq $Guid }
                        }
                        else
                        {
                            $WhereBlock = { $_.GetValue('DisplayName') }
                        }
                        $SwKeys = Get-ChildItem -Path $UninstallKey -
ErrorAction SilentlyContinue | Where-Object $WhereBlock
                        if (-not $SwKeys)
                        {
                            Write-Verbose -Message "No software keys in
uninstall key $UninstallKey"
                        }
                        else
                        {
                            foreach ($SwKey in $SwKeys)
                            {
                                $output = @{ }
                                foreach ($ValName in $SwKey.GetValueNames())
                                {
                                    if ($ValName -ne 'Version')
                                    {
                                        $output.InstallLocation = ''
```

```
                                        if ($ValName -eq 'InstallLocation' -
and ($SwKey.GetValue($ValName)) -and (@('C:', 'C:\Windows',
'C:\Windows\System32', 'C:\Windows\SysWOW64') -notcontains
$SwKey.GetValue($ValName).TrimEnd('\')))
                                        {
                                            $output.InstallLocation =
$SwKey.GetValue($ValName).TrimEnd('\')
                                        }
                                        [string]$ValData =
$SwKey.GetValue($ValName)
                                        if ($friendlyNames[$ValName])
                                        {
$output[$friendlyNames[$ValName]] = $ValData.Trim() ## Some registry values
have trailing spaces.
                                        }
                                        else
                                        {
                                            $output[$ValName] =
$ValData.Trim() ## Some registry values trailing spaces
                                        }
                                    }
                                }
                                $output.GUID = ''
                                if ($SwKey.PSChildName -match '\b[A-
F0-9]{8}(?:-[A-F0-9]{4}){3}-[A-F0-9]{12}\b')
                                {
                                    $output.GUID = $SwKey.PSChildName
                                }
                                New-Object –TypeName PSObject –Prop $output
                            }
                        }
                    }
                }
            }
            if ($ComputerName -eq $env:COMPUTERNAME)
            {
                & $scriptBlock $PSBoundParameters
            }
            else
            {
                Invoke-Command -ComputerName $ComputerName -ScriptBlock
$scriptBlock -ArgumentList $PSBoundParameters
            }
        }
        catch
        {
            Write-Error -Message "Error: $($_.Exception.Message) - Line
Number: $($_.InvocationInfo.ScriptLineNumber)"
        }
    }
}
```