Trabajar con el registro desde PowerShell

En este artículo veremos cómo desde PowerShell podemos trabajar con el registro utilizando los cmdlets propios de PowerShell, así como personalizaremos tareas definiendo funciones.

Contenidos

El Proveedor `Registry`

Para trabajar con el registro desde PowerShell se utiliza el proveedor Registry. **Con el proveedor `Registry`, podemos buscar, crear, modificar y eliminar valores y claves, así como administrar las *ACLs*** (Access Control List), los permisos de las claves, **todo ello con la comodidad de hacerlo como si del sistema de archivos se tratara**.

Las Rutas Del Registro Y Los Drives Del Proveedor `Registry`

Para hacer referencia a las claves y valores es necesario pasar sus rutas al proveedor registry. Esto se puede hacer de dos maneras, **usando las rutas en sí mismas o usando los Drives definidos en PowerShell**.

El proveedor Registry tiene dos Drives: `hklm:` (HKEY_LOCAL_MACHINE) y `hkcu:` (HKEY_CURRENT_USER). Podemos situarnos en cualquiera de ellos por medio del comando Set-Location (o su alias `cd`):

```
PS> Set-Location hklm:
PS> cd hkcu:
```

Los Drives `hklm:` y `hkcu:` nos facilitan el acceso al registro, pero como dijimos antes, **no son la única manera de poner una ruta del registro**. Si quisieramos ir al perfil predeterminado, tendríamos que **poner como ruta la palabra clave `registry` seguido de dos dobles puntos y la ruta,** ya sea **con el nombre completo de la clave raíz o usando su abreviatura** (Las abreviaturas de las claves raíces del registro son `HKCR` = `HKEY_CLASSES_ROOT`, `HKCU` = `HKEY_CURRENT_USER`, `HKLM` = `HKEY_LOCAL_MACHINE`, `HKU` = `HKEY_USERS` y `HKCC` = `HKEY_CURRENT_CONFIG`). En los siguientes ejemplos nos situaremos en la raíz del perfil del usuario predeterminado:

```
PS> Set-Location registry::HKEY_USERS\.Default
PS> Set-Location registry::HKU\.Default
```

Obtener Claves Y Valores

De manera general, podemos decir que **para listar claves se utiliza Get-ChildItem,** y **para obtener el contenido de los valores Get-ItemProperty**.

Obtener Claves

Una vez situados en una clave, podemos listar las claves que contiene con Get-ChildItem y navegar por ellas, como si de un árbol de directorios de disco se tratara, con Set-Location:

```
PS> Set-Location hkcu:
PS> Get-ChildItem

   Hive: Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER

SKC  VC Name                           Property
---  -- ----                           --------
  2   0 AppEvents                     {}
  0  33 Console                      {ColorTable00, ColorTable01, ColorTable02, ColorTable03...}
 24   1 Control Panel                 {Opened}
  0   3 Environment                   {Path, TEMP, TMP}
  1   6 Identities                    {Identity Ordinal, Migrated5, Last Username, Last User ID...}
  3   0 Keyboard Layout               {}
  0   0 Network                       {}
  3   0 Printers                      {}
 22   0 Software                      {}
  0   0 UNICODE Program Groups        {}
  2   0 Windows 3.1 Migration Status  {}
```

Vemos aquí listadas las claves contenidas en HKEY_CURRENT_USER. **La columna SKC nos indica cuántas subclaves contiene la clave** (dos en el caso de AppEvents, ninguna en Console), **la columna VC nos indica cuántos valores hay contenidos en la clave** (ninguno en AppEvents y 33 en Console), **la columna Name el nombre de la clave** listada **y la columna Property los nombres de los valores que contiene** (lista truncada, al ser la salida en formato tabla, cuando se supera el ancho de consola).

Obtener Valores

Para obtener los valores contenidos en una clave usamos Get-ItemProperty y la ruta de la clave:

```
PS> Get-ItemProperty hkcu:\Console

PSPath : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\Console
PSParentPath : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER
PSChildName : Console
```

```
PSDrive : HKCU
PSProvider : Microsoft.PowerShell.Core\Registry
ColorTable00 : 0
ColorTable01 : 8388608
ColorTable02 : 32768
ColorTable03 : 8421376
ColorTable04 : 128
ColorTable05 : 8388736
ColorTable06 : 32896
ColorTable07 : 12632256
ColorTable08 : 8421504
ColorTable09 : 16711680
ColorTable10 : 65280
ColorTable11 : 16776960
ColorTable12 : 255
ColorTable13 : 16711935
ColorTable14 : 65535
ColorTable15 : 16777215
CursorSize : 25
FaceName :
FontFamily : 0
FontSize : 0
FontWeight : 0
FullScreen : 0
HistoryBufferSize : 50
InsertMode : 1
LoadConIme : 1
NumberOfHistoryBuffers : 4
PopupColors : 179
QuickEdit : 0
ScreenBufferSize : 98304120
ScreenColors : 62
WindowSize : 3276920
CurrentPage : 3
HistoryNoDup : 0
```

Como podemos ver, Get-ItemProperty no sólo nos devuelve los valores contenidos en la clave, si no que también devuelve cinco propiedades de la clave en sí (`PSParent`, `PSParentPath`, `PSChildName`, `PSDrive` y `PSProvider`). Si queremos listar sólo los valores sin estas cinco propiedades, el procedimiento es algo más complejo, usandose Get-Item con la ruta de la clave y canalizando la salida a Select-Object, al que se pasa `Property` como valor al parámetro `-ExpandProperty`; este mecanismo nos permite listar los elementos del array `Property`, que son los nombres de los valores que contiene la clave:

```
PS> Get-Item hkcu:\Console | Select-Object -ExpandProperty Property
ColorTable00
ColorTable01
ColorTable02
ColorTable03
ColorTable04
ColorTable05
ColorTable06
ColorTable07
ColorTable08
ColorTable09
ColorTable10
ColorTable11
ColorTable12
ColorTable13
ColorTable14
ColorTable15
CursorSize
FaceName
FontFamily
FontSize
FontWeight
FullScreen
HistoryBufferSize
InsertMode
LoadConIme
NumberOfHistoryBuffers
PopupColors
QuickEdit
ScreenBufferSize
ScreenColors
WindowSize
CurrentPage
HistoryNoDup
```

Para obtener el contenido de un valor en concreto, usamos Get-ItemProperty con el modificador `-name` y el nombre del valor y canalizamos a Format-List pasando el nombre del valor a este comando:

```
PS> Get-ItemProperty hkcu:\Console -name ColorTable01|Format-List ColorTable01

ColorTable01 : 8388608
```

Podemos crear una función que nos simplifique esta labor:

```
PS> function Obtener-ValorDelRegistro{Get-ItemProperty $args[0] -name $args[1] | Format-List $args[1]}
PS> Obtener-ValorDelRegistro hkcu:\Console ColorTable01

ColorTable01 : 8388608
```

Hay otra manera de obtener esto mismo sin necesidad de crear ninguna función. Consiste en invocar Get-ItemProperty, pasando a `-Path` la ruta de la clave, todo ello encerrado entre paréntesis y precedido del caracter de dólar, seguido de un punto y del nombre del valor a consultar. En el siguiente ejemplo obtenemos el contenido del valor `Principios` que está en la clave `HKEY_CURRENT_USER\Moral`:

```
PS> $(Get-ItemProperty -Path registry::hkcu\Moral).Principios
Estos son mis principios, si no le gustan tengo otros
```

En conjunción con lo visto anteriormente, podemos crear una función que nos enumere todos los valores y su contenido de una clave, basada en la enumeración realizada con Get-Item y Select-Object y la consulta a un determinado valor que hemos visto ahora mismo:

```
function Obtener-ValoresDeClave($clave)
{
    $valores = Get-Item $clave | Select-Object -ExpandProperty Property
    foreach ($Valor in $Valores)
```

```
    {
        Get-ItemProperty $Clave -name $Valor | Format-List $Valor
    }
}
```

Veamos un ejemplo de llamada a la función:

```
PS> Obtener-ValoresDeClave hkcu:\Console
```

Para consultar el valor predeterminado de una clave, debemos pasar como nombre `(default)`, pero esto da error al ser ignorados los paréntesis por ser considerados como englobadores; para conseguir que sean tomados como literales, debemos precederles del caracter de acento grave (`). Si no se ha establecido el valor predeterminado en una clave, este comando da error:

```
PS> Get-ItemProperty hkcu:/Console/MiClave -name `(default`) |Format-List `(default`)

(default) : Este es el valor
```

Al igual que antes, podemos crear una función que nos simplifique esta labor:

```
PS> function Obtener-ValorPredeterminado{Get-ItemProperty $args[0] -name `(default`) | Format-List `(default`)}
PS> Obtener-ValorPredeterminado hkcu:/Console/MiClave

(default) : Este es el valor
```

Creación De Claves Y Valores

Para la creación de claves se utiliza New-Item y para la creación de valores New-ItemProperty.

Creación De Claves

Para crear una clave usamos New-Item, pasándole con `-Path` la ruta de la clave en la que está contenida y co `-Name` el nombre:

```
PS> New-Item -path hkcu:\Console -Name MiClave
```

También **podemos crearla asignando contenido al valor predeterminado, usando el parámetro `-value`**. Si como contenido se pasa un array de bytes, el valor será de tipo *REG_BINARY*; cualquier otro tipo de contenido será tomado como *REG_SZ*:

```
PS> New-Item -path hkcu:/Console -Name MiClaveREG_SZ -value "Este es el valor predeterminado de MiClaveREG_SZ"
PS> $ArrayDeBytes = [byte[]](68,120,50,49,44,160,76,76,67)
PS> New-Item -path hkcu:\Console -Name MiClaveREG_BINARY -value $ArrayDeBytes
```

Cuando desde el editor del registro (regedit) se crea una clave, ésta tiene el valor predeterminado vacío y de tipo *REG_SZ*. Usando New-Item y el modificador `-itemType`, podemos crear la clave especificando otros tipos para el valor predeterminado. Los tipos posibles son `String` para *REG_SZ*, `ExpandString` para *REG_EXPAND_SZ*, `Binary` para *REG_BINARY*, `DWord` para *REG_DWORD*, `MultiString` para *REG_MULTI_SZ*, `QWord` para *REG_QWORD* y `Unknown` para tipo no especificado (el resultado es que se crea un valor predeterminado de tipo *REG_SZ* o *REG_BINARY*, según el dato pasado, al igual que sucede cuando no se especifica el tipo con `-type`). Vemos a continuación la creación de claves con el valor predeterminado de cada uno de los distintos tipos:

```
PS> New-Item -path hkcu:\Console -Name MiClaveREG_SZ -itemType String -value "Este es el valor predeterminado de MiClaveREG_SZ"
PS> New-Item -path hkcu:\Console -Name MiClaveREG_EXPAND_SZ -itemType ExpandString -value "%COMPUTERNAME%"
PS> New-Item -path hkcu:\Console -Name MiClaveREG_MULTI_SZ -itemType MultiString -value "Linea1`r`nLinea2`r`nLinea3"
PS> New-Item -path hkcu:\Console -Name MiClaveREG_BINARY -itemType Binary -value $ArrayDeBytes
PS> New-Item -path hkcu:\Console -Name MiClaveREG_DWORD -itemType DWord -value 6969
PS> New-Item -path hkcu:\Console -Name MiClaveREG_QWORD -itemType QWord -value 6969696969
PS> New-Item -path hkcu:\Console -Name MiClaveUnknownREG_SZ -itemType Unknown -value "Este es el valor predeterminado de MiClaveUnknownREG_SZ"
PS> New-Item -path hkcu:\Console -Name MiClaveUnknownREG_BINARY -itemType Unknown -value $ArrayDeBytes
```

Creación De Valores

Para crear un valor dentro de una clave usamos New-ItemProperty, pasando la ruta de la clave en la que está contenido el valor con `-Path`, el tipo de valor con `-propertyType`, el nombre con `-Name` y el contenido con `-Value`. Los tipos de valores especificados con `-propertyType` son los mismos que vimos antes (`String`, `MultiString`, `ExpandString`, `Binary`, `DWord`, `QWord` y `Unknown`) y de igaul modo al visto con anterioridad, si no se especifica el tipo se creará *REG_BINARY* o *REG_SZ* según reciba un array de bytes o cualquier otro dato:

```
PS> New-ItemProperty -path hkcu:/Console/MiClave -propertyType String -Name Tipo-String -value "Este es el valor REG_SZ"
PS> New-ItemProperty -path hkcu:/Console/MiClave -propertyType ExpandString -Name Tipo-ExpandString -value "%COMPUTERNAME%"
PS> New-ItemProperty -path hkcu:/Console/MiClave -propertyType MultiString -Name Tipo-MultiString -value "Linea1`r`nLinea2`r`nLinea3"
PS> New-ItemProperty -path hkcu:/Console/MiClave -propertyType Binary -Name Tipo-Binary -value $ArrayDeBytes
PS> New-ItemProperty -path hkcu:/Console/MiClave -propertyType DWord -Name Tipo-DWord -value 6969
PS> New-ItemProperty -path hkcu:/Console/MiClave -propertyType QWord -Name Tipo-QWord -value 6969
PS> New-ItemProperty -path hkcu:/Console/MiClave -propertyType Unknown -Name Tipo-UnknownString -value "Este es el valor Unknown REG_SZ"
PS> New-ItemProperty -path hkcu:/Console/MiClave -propertyType Unknown -Name Tipo-UnknownBinary -value $ArrayDeBytes
PS> New-ItemProperty -path hkcu:/Console/MiClave -Name Tipo-OtroString -value "Este es el valor Otro REG_SZ"
PS> New-ItemProperty -path hkcu:/Console/MiClave -Name Tipo-OtroBinary -value $ArrayDeBytes
```

Por cierto, que tanto con New-Item como con New-ItemProperty se puede usar `-Type` (en lugar de `-itemType` o `-propertyType`) para especificar el tipo de dato, pues *PowerShell* es capaz de autocompletar comandos y opciones siempre que lo que se le ponga no pueda ser también parte del nombre de otro parámetro.

Copiar Claves Y Valores

Para copiar claves se utiliza Copy-Item y para copiar valores Copy-ItemProperty.

Copiar Claves

Podemos copiar una clave completa (con todos sus valores y subclaves) en otra, de la misma forma que se hace con las carpetas del sistema de archivos, **usando Copy-Item, `-path` para la ruta de la clave a copiar, `-destination` para la clave dentro de la que se copiará** (si especificamos al final de la ruta un nombre de clave que no existe, se copiará cambiando el nombre de la clave a dicho nombre) **y `-recurse` para que copie subclaves y valores**. En los dos ejemplos siguientes copiamos la clave con su nombre, en el primero y la clave con otro nombre en el segundo:

```
PS> Copy-Item -path hkcu:/Console/MiClave -recurse -destination hkcu:/Console/MiClaveREG_BINARY
PS> Copy-Item -path hkcu:/Console/MiClave -recurse -destination hkcu:/Console/MiClaveREG_BINARY/MiClaveCopia
```

Copiar Valores

Para copiar un valor se utiliza Copy-ItemProperty, -path para la ruta de la clave que lo contiene, -destination para la ruta de la clave donde se copiará y -name para el nombre del valor a copiar.

```
PS> Copy-ItemProperty -Path hkcu:\Console\MiClave -Destination hkcu:\Console\MiClaveCopia -Name Valor1
```

Copiar los valores de una clave en otra usando Copy-ItemProperty no es algo que se pueda hacer directamente, sería necesario enumerarlas una a una e ir creando las nuevas como copia de las anteriores. Esto podría ser realizado con una función:

```
function Copiar-ValoresDeClave($Origen,$Destino)
{
```

```
    $Valores = (Get-Item $Origen | Select-Object -ExpandProperty Property)
    foreach($Valor In $Valores)
    {
        Copy-ItemProperty -Path $Origen -Destination $Destino -Name $Valor
    }
}
```

Veamos un ejemplo de uso de esta función:

```
PS> copiar-valoresdeclave -Origen HKCU:\Origen -Destino HKCU:\Destino
```

En el anterior ejemplo, los valores de la clave HKEY_CURRENT_USER\Origen son copiados en la clave HKEY_CURRENT_USER\Destino.

Renombrar Claves Y Valores

Para renombrar claves se utiliza Rename-Item y para renombrar valores Rename-ItemProperty.

Renombrar Claves

Otra tarea que podemos querer realizar en el registro es la de renombrar valores y claves. **Para renombrar una clave se utiliza el cmdlet Rename-Item, con `-Path` para establecer ruta de la clave a renombrar y `-NewName` para la nueva ruta**:

```
PS> Rename-Item -path hkcu:/Console/MiClave -NewName hkcu:/Console/MiClaveNueva
```

Renombrar Valores

Para renombrar un valor utilizamos el cmdlet Rename-ItemProperty, pasando `-Path` para la clave que lo contiene, `-Name` para el nombre del valor y `-NewName` para el nuevo nombre:

```
PS> Rename-ItemProperty -path hkcu:/Console/MiClaveNueva -Name Valor1 -NewName NuevoValor1
```

Vaciar El Contenido De Claves Y Valores

Para vaciar el contenido de una clave se utiliza Clear-Item, para vaciar el contenido de un valor Clear-ItemProperty.

Vaciar Claves

Podemos vaciar una clave de valores, con el cmdlet Clear-Item, al que se pasa `-Path` para la clave a vaciar:

```
PS> Clear-ItemProperty -path hkcu:/Console/MiClaveNueva
```

El anterior comando sólo vacía los valores contenidos en la clave especificada como parámetro `-Path`. Si esta clave contiene subclaves, estas subclaves quedan intactas (no son borradas), así como los valores que contengan. Si quiséramos vaciar todos los valores de todas las subclaves, deberíamos crear una función que lo hiciera:

```
Function Vaciar-Clave($Clave)
{
    foreach($SubClave In Get-ChildItem $Clave)
    {
        $Ruta = "registry::" + $SubClave.Name
        s_VaciarClave -Clave $Ruta
    }
    Clear-Item -Path $Clave
}
```

Veamos un ejemplo de llamada a esta función:

```
PS> Vaciar-Clave -Clave registry::HKCU\Console\MiClave
```

Si quisieramos que se eliminasen también las subclaves contenidas en la clave que estamos vaciando, tendríamos que hacer otra función, que usase Remove-Item para eliminar las subclaves y Clear-Item para eliminar los valores:

```
Function Vaciar-Clave($Clave)
{
    foreach($SubClave In Get-ChildItem $Clave)
    {
        $Ruta = "registry::" + $SubClave.Name
        Remove-Item -Path $Ruta -Recurse
    }
    Clear-Item -Path $Clave
}
```

Veamos un ejemplo de llamada a esta función:

```
PS> Vaciar-Clave -Clave registry::HKCU\Console\MiClave
```

Las dos funciones anteriores se podrían englobar en una sola, que recibiera como argumento opcional un modificador que establezca si se eliminan o no las subclaves, siendo su valor predeterminado que no se eliminen:

```
Function Vaciar-Clave($Clave,$Recursivo = $false)
{
    foreach($SubClave In Get-ChildItem $Clave)
    {
        $Ruta = "registry::" + $SubClave.Name
         If($Recursivo -eq $false)
         {
             s_VaciarClave -Clave $Ruta
        }
        Else
        {
            Remove-Item -Path $Ruta -Recurse
        }
    }
    Clear-Item -Path $Clave
}
```

Vemos tres ejemplos de la llamada a esta función:

```
PS> Vaciar-Clave -Clave registry::HKCU\Console\MiClave #vacía los valores de la clave y de las subclaves
PS> Vaciar-Clave -Clave registry::HKCU\Console\MiClave -Recursivo $false #vacía los valores de la clave y de las subclaves
PS> Vaciar-Clave -Clave registry::HKCU\Console\MiClave -Recursivo $true #vacía los valores de la clave y borra las subclaves
```

Vaciar Valores

Podemos quitar el contenido a un valor con Clear-ItemProperty, pasando `-Path` para la clave en la que está contenido y `-Name` para su nombre. Esto

implica poner cadena vacía en los valores de cadena y cero en los valores numéricos:

```
PS> Rename-ItemProperty -path hkcu:/Console/MiClaveNueva -Name NuevoValor1
```

Mover Claves Y Valores

Para mover claves se utiliza Move-Item y para mover valores Move-ItemProperty.

Mover Claves

Se puede mover una clave, sus valores y todas las subclaves que contenga utilizando el cmdlet Move-Item, pasándole `-Path` como la ruta de la clave a mover y `-Destination` como la ruta de la clave a la que se moverá:

```
PS> Move-Item -Path hkcu:/Console/MiClave -Destination hkcu:/Console #mueve la clave a la raíz de HKCU
```

Mover Valores

Para mover un valor se usa el cmdlet Move-ItemProperty, pasándole `-Path` para la clave que lo contiene, `-Destination` para la clave a la que se moverá y `-Name` para el nombre del valor:

```
PS> Move-ItemProperty -Path hkcu:\Console\MiClave -Destination hkcu:\Console\MiNuevaClave -Name Valor1
```

Para mover todos los valores contenidos en una clave, sin que ésta sea borrada, necesitamos crear una función. Podríamos hacerlo con una función que copiara la clave al completo con Copy-Item en la nueva ubicación y despues vaciase la clave de origen utilizando Clear-Item para borrar los valores y Remove-Item para borrar las subclaves; esto último lo podemos hacer con una llamada a la función `Vaciar-Clave` que hemos visto anteriormente:

```
Function Mover-Clave($Origen,$Destino)
{
    Copy-Item -Path $Origen -Destination $Destino -Recurse
    Vaciar-Clave -Clave $Origen -Recursivo $true
}
```

Veamos un ejemplo de llamada a esta función:

```
PS> Mover-Clave -Origen hkcu:\Origen -Destino hkcu:\Destino
```

Cambiar El Contenido De Los Valores

Para cambiar el contenido de un valor, usamos Set-ItemProperty, pasándole `-Path` para la ruta de la clave que lo contiene, `-Name` para el nombre del valor y `-Value` para el valor a establecer:

```
PS> Set-ItemProperty -Path hkcu:\Console\MiClave -Name Cadena -Value "Valor de cadena"
PS> Set-ItemProperty -Path hkcu:\Console\MiClave -Name CadenaExpandible -Value "%DATE% - Valor de cadena expandible"
PS> Set-ItemProperty -Path hkcu:\Console\MiClave -Name CadenaMultiple -Value ([string[]]("Valor de cadena 1","Valor de cadena 2"))
PS> Set-ItemProperty -Path hkcu:\Console\MiClave -Name Binario -Value ([byte[]](1,2,3,4,5))
PS> Set-ItemProperty -Path hkcu:\Console\MiClave -Name DWORD -Value 0x69 #expresado en hexadecimal
PS> Set-ItemProperty -Path hkcu:\Console\MiClave -Name DWORD -Value 105 #expresado en decimal
```

Debemos destacar como hemos pasado el dato de binario y de cadena múltiple. En ambos casos se trata de valores de tipo array, uno de cadenas y el otro de bytes. Por ello ponemos los valores como array, es decir, separados por comas, entre comillas las cadenas y sin ellas los números; a estos valores les forzamos el tipo por medio de string o byte, según el caso, indicando que además es un array por medio de los dobles corchetes.

Borrar Claves Y Valores

Para borrar claves se utiliza Remove-Item y para borrar valores Remove-ItemProperty.

Borrar Claves

Para borrar una clave, se utiliza el cmdlet Remove-Item, pasándole `-Path` para el nombre de la clave a borrar:

```
PS> Remove-Item -Path hkcu:\Console\MiClaveABorrar
```

Esto provoca que se borre la clave y todos sus valores. **Si la clave contiene subclaves, se pedirá que se confirme el borrado de las mismas**:

```
PS HKCU:\> remove-item -path hkcu:\Console\MiClaveABorrar

Confirmar
El elemento situado en HKCU:\Console\MiClaveABorrar tiene elementos secundarios y no se especificó el
parámetro Recurse. Si continúa, se quitarán todos los secundarios junto con el elemento. ¿Está seguro
de que desea continuar?
[S] Sí [O] Sí a todo [N] No [T] No a todo [U] Suspender [?] Ayuda (el valor predeterminado es "S"
```

Para evitar esto, tal y como indica el propio mensaje, **se debe pasar el modificador `-Recurse`, con lo que se borrarán las subclaves sin pedir confirmación**:

```
PS HKCU:\> remove-item -path hkcu:\Console\MiClaveABorrar -Recurse
```

Borrar Valores

Para borrar un valor se utiliza el cmdlet Remove-ItemProperty, pasando `-Path` para el nombre de la clave donde está contenido y `-Name` para su nombre:

```
PS> Remove-ItemProperty -Path hkcu:\Console\MiClave -Name Valor1
```

Conclusión

En el presente artículo hemos visto el manejo de las claves y los valores en PowerShell, realizando desde línea de comando consultas a los valores y claves, modificaciones, creación, vaciado y eliminación de los mismos. Hemos visto cómo con PowerShell podemos crear funciones que nos permitan realizar estas tareas de una manera más cómoda y todo nos da una idea las enormes posibilidades que PowerShell nos oferta a la hora de crear scripts que automaticen estas tareas con el registro.

Para otro artículo queda el tema de trabajar con la seguridad de las claves, pero es "otra historia" -(|:o)).