

# Active Directory Domain Services

Aunque podemos administrar los objetos contenidos en AD DS con el proveedor de servicios «WINNT» es preferible utilizar el proveedor «LDAP» creado especialmente para ello.

En este apartado vamos a descubrir como administrar las Unidades Organizativas (UO), los diferentes tipos de grupos, y por supuesto los usuarios de dominio.

## 1. Conexión al servicio de directorio y a sus objetos

Debemos en primer lugar descubrir cómo conectarse a los objetos antes de poderlos administrar. La conexión a un objeto se efectúa con la ayuda del protocolo LDAP, donde debemos especificar el DN (*Distinguished Name*) de nuestro objeto.

El DN permite identificar de forma única un objeto en una jerarquía. Un DN es por tanto un nombre único que representa una «ruta» hacia un objeto situado en un servicio de directorio.

### Conexión a un usuario

Veamos un ejemplo de DN que identifica un objeto usuario en el seno del dominio «PS-Scripting.com»:

```
LDAP://CN=oscar,OU=usuarios,DC=ps-scripting,DC=com
```

El *Distinguished Name* se compone del CN (*Common Name*) es decir, el nombre del objeto, seguido de la UO que contiene el objeto, seguido también del nombre del dominio. En cuando a este último, cada componente del dominio debe especificarse utilizando la notación «DC=» (DC, por *Domain Component*).

El DN se construye empezando por el CN del objeto y remontando la jerarquía del dominio hasta llegar a la raíz.

Si nuestro objeto usuario está contenido en una jerarquía de unidades organizativas, podríamos construir el DN de la manera siguiente:

```
LDAP://CN=oscar,OU=Info,OU=España,DC=ps-scripting,DC=com
```

En este ejemplo, el usuario «oscar» se sitúa en la UO «Info» la cual se sitúa en la UO «España».

Ahora que ya sabemos cómo crear un DN, debemos pasarlo a nuestro adaptador de tipo [ADSI] para poder conectarnos a un objeto. Esto se realiza de la siguiente forma:

```
PS > $objUser=[ADSI]'LDAP://CN=oscar,OU=Info,OU=España,DC=ps-scripting,DC=com'
```

Ahora que estamos conectados a un objeto, vamos a poder manipularlo o simplemente observar sus propiedades. Para verificar la conexión, pruebe de mostrar la variable \$objUser, como se indica a continuación:

```
PS > $objUser

distinguishedName
-----
{CN=oscar,OU=Info,OU=España,DC=ps-scripting,DC=com}
```

Si el DN se muestra como en el ejemplo, querrá decir que nuestra consulta LDAP tuvo éxito y que es válido. En caso contrario, obtenemos un mensaje de error.

### Conexión a la raíz del dominio actual

Acabamos de ver cómo conectarnos a un usuario del que conocemos de antemano su DN, pero no siempre es este el caso. Así, gracias a una consulta LDAP apropiada será posible buscar un objeto en el servicio de directorio. Pero antes de

llegar a eso, es necesario que nos conectemos a la raíz del servicio de directorio.

Para ello, hemos visto en el ejemplo anterior que nos basta con especificar el nombre del dominio en el formato `DC=MiSubDominio,DC=MiDominio`. De este modo podremos escribir lo siguiente:

```
$objDominio=[ADSI]'LDAP://DC=ps-scripting,DC=com'
```

En efecto funciona, pero ello implica que conocíamos previamente, una vez más, el nombre del dominio. Existe otra técnica, mucho más trivial pero que hace la misma cosa. Ahora escribiremos esto:

```
$objDominio=[ADSI]''
```

Para verificar que estamos bien conectados, intentemos mostrar el contenido de `$objDominio`:

```
PS > $objDominio

distinguishedName
-----
{DC=ps-scripting,DC=com}
```

Esta sintaxis tiene la ventaja no solamente de la concisión, sino sobretodo que aporta una cierta independencia respecto al AD DS; nos asegurará por tanto una mejor portabilidad de nuestros scripts.

### **Conexión a un catálogo global**

Para ampliar el perímetro de búsqueda a todos los usuarios, grupos globales y universales del bosque, podemos también conectarnos a un catálogo global mediante el prefijo «GC://».

```
$objGC=[ADSI]'GC://DC=ps-scripting,DC=com'
```

Como puede ver lo que tenemos que hacer es sustituir «LDAP» por «GC» en nuestra petición de conexión. Ésta nos permitirá conectar a un catálogo global y comunicarnos con este último por el puerto TCP 3268 en lugar de 389 (utilizado por LDAP).



Recuerde que si se conecta a un catálogo global tendrá acceso a más objetos del servicio de directorio (todos los del bosque); pero estos objetos pueden tener menos atributos que cuando consultó al controlador de dominio que los alberga. Esto es así, principalmente, para las cuentas de usuario.

### **Conexión a un dominio remoto bajo otra identidad**

Cuando usamos el atajo [ADSI], PowerShell nos simplifica en realidad el acceso a las clases del Framework .NET. Así si observamos con más detalle el tipo de nuestra variable `$objDominio`, constataremos que se trata de un objeto de tipo `System.DirectoryServices.DirectoryEntry`.

Pasando por la notación [ADSI] no podemos sacar partido de todas las funcionalidades de la clase `DirectoryEntry`, como la que permite la «*impersonación*». La *impersonación* es un mecanismo que permite la ejecución de código bajo la identidad de otro usuario. Tendremos acceso haciendo la llamada directamente al Framework .NET.

Por este medio, vamos a poder hacer las dos operaciones siguientes:

- conectarnos a un dominio distinto de aquel en que se encuentra nuestra máquina,
- conectarnos al dominio de nuestra máquina bajo otra identidad.

El primer punto es muy importante, ya que puede disponer de múltiples dominios en su organización y querer realizar funciones de administración en el dominio A mientras su máquina se encuentra en el dominio B. Esto también se aplicará si su máquina se encuentra en un *workgroup*, dicho de otro modo, fuera de cualquier dominio.

Para ello, utilizamos el trozo de código siguiente:

```
$objDominio = New-Object System.DirectoryServices.DirectoryEntry(
'LDAP://Nombre_o_IP_Dominio', 'CuentaAdmin', 'contraseña')
```

Nombre\_o\_IP\_Dominio: deberá especificar aquí, o bien un nombre de dominio completo (FQDN (*Fully Qualified Domain Name*)) (por ejemplo ps-scripting.com), o bien la dirección IP de un controlador de dominio.

#### Ejemplo:

```
PS > $objDominio = New-Object System.DirectoryServices.DirectoryEntry(
'LDAP://192.160.1.1', 'administrador', 'P@ssw0rd')
```

Ahora que estamos conectados, sólo nos queda definir las acciones que se desea realizar en los objetos del AD DS. Esto es lo que vamos a ver en el apartado siguiente...

Acabamos de ver las diferentes formas de conectarse a AD DS: a través de un atajo [ADSI], o bien utilizando la clase `DirectoryEntry` del Framework .NET. Usted tendrá que elegir la que mejor se ajuste a sus necesidades en función de su contexto de trabajo y de lo que quiera hacer.



Para todos los ejemplos siguientes, siempre haremos como si administrásemos el dominio actual. Es decir el dominio en el que se encuentra nuestra máquina y a partir de la cual ejecutaremos nuestros comandos. Supondremos también que estamos conectados con una cuenta con derechos del administrador del dominio.

## 2. Búsqueda de objetos

Gracias al Framework .NET y particularmente a la clase `DirectorySearcher` vamos a poder efectuar potentes búsquedas en el servicio de directorio Active Directory. Cuando se instancie un objeto `DirectorySearcher`, debemos obligatoriamente pasarle por parámetro la ruta a partir de la cual tendrá lugar la búsqueda. Opcionalmente y para restringir la búsqueda, podemos indicar un filtro de búsqueda, así como una lista de propiedades a buscar.

También disponemos de un gran conjunto de propiedades, veamos aquí un extracto de las más usuales:

Propriedad	Tipo	Descripción
CacheResults	Boolean	Especifica si el resultado de la búsqueda debe ser o no almacenada en la caché del equipo. Valor por <b>defecto True</b> . Es aconsejable poner el valor a <b>Falso</b> si la búsqueda devuelve un gran número de objetos.
ServerTimeLimit	TimeSpan	Tiempo máximo asignado a la búsqueda. Por <b>defecto</b> 120 segundos ( <b>valor -1</b> ). Cuando el límite de tiempo se alcanza, la búsqueda se interrumpirá y el servidor devolverá los resultados obtenidos.
Filter	String	Define un filtro en formato LDAP del tipo «(objectClass=user)». El filtro aplicado por <b>defecto</b> es «(objectClass=*)». Es posible crear filtros elaborados basados en los operadores & (Y) y   (O). Ejemplo: (& (objectClass= user) (lastName=Davis)). Los paréntesis son obligatorios.
SearchScope	String	Ámbito de la búsqueda. La búsqueda puede restringirse a la UO especificada (valor <b>Base</b> ), a un solo nivel de profundidad (valor <b>OneLevel</b> ), a todos los sub contenedores (valor <b>SubTree</b> ). <b>Defecto: SubTree</b> .
PageSize	Integer	Número de resultados a devolver por páginas de búsqueda. Por <b>defecto</b> , PageSize vale <b>zero</b> , lo que significa que este mecanismo no está activo. Por consiguiente, el número de resultados devueltos no puede exceder de

		<b>1000.</b> Para superar este límite deberá indicar un valor superior a cero para esta propiedad.
PropertiesToLoad	String	Conjunto de propiedades a recuperar. Por <b>defecto todas</b> las propiedades se recuperarán.

Finalmente, dispondremos de los dos métodos siguientes:

**FindOne:** ejecuta la consulta y retorna el primer resultado encontrado.

**FindAll:** ejecuta la consulta y retorna una colección que contendrá los objetos encontrados.

Ahora, probaremos una pequeña búsqueda. Comprobamos estas líneas de código:

```
PS > $objDominio = [ADSI]''
PS > $objBusqueda =
    New-Object System.DirectoryServices.DirectorySearcher($objDominio)

PS > $objBusqueda

CacheResults           : True
ClientTimeout          : -00:00:01
PropertyNamesOnly      : False
Filter                 : (objectClass=*)
PageSize               : 0
PropertiesToLoad        : {}
ReferralChasing        : External
SearchScope            : Subtree
ServerPageTimeLimit    : -00:00:01
ServerTimeLimit        : -00:00:01
SizeLimit              : 0
SearchRoot             : System.DirectoryServices.DirectoryEntry
Sort                   : System.DirectoryServices.SortOption
Asynchronous           : False
Tombstone              : False
AttributeScopeQuery    :
DerefAlias             : Never
SecurityMasks          : None
ExtendedDN             : None
DirectorySynchronization :
VirtualListView         :
Site                   :
Container              :
```

Acabamos de crear un objeto `DirectorySearcher` y hemos mostrado sus propiedades. Obtendremos las propiedades y sus valores por defecto que hemos detallado en el cuadro anterior.

Ahora, para ejecutar nuestra consulta, debemos utilizar uno de las dos métodos `FindOne` o `FindAll`.

Probemos el método `FindOne`:

```
PS > $objBusqueda.FindOne() | Format-List

Path           : LDAP://ps-scripting.com/DC=powershell-scripting,DC=com
Properties      : {fsmoroleowner, minpwdlength, adspath, msds-perusertrustto...}
```

## a. Obtener la lista de las unidades organizativas

Puesto que ya controlamos perfectamente la búsqueda, obtener una lista de OU será como un juego de niños. Como de costumbre, primero debemos conectarnos a Active Directory. La conexión debe efectuarse en función de la consulta que se desea hacer. ¿Debemos listar todas las OU del AD DS o sólo las contenidas en una OU especial? La respuesta a esta pregunta será la que determinará la constitución de la petición de conexión al servicio de directorio.

Listemos todas las UO a partir de la raíz:

```
PS > $objDominio = [ADSI]''
PS > $objBusqueda =
    New-Object System.DirectoryServices.DirectorySearcher($objDominio)

PS > $objBusqueda.Filter='(objectCategory=organizationalUnit)'
PS > $objBusqueda.FindAll() | Format-List

Path          : LDAP://ps-scripting.com/OU=Domain Controllers,
                DC=ps-scripting,DC=com
Properties    : {ou, name, whencreated, iscriticalsystemobject...}

Path          : LDAP://ps-scripting.com/OU=Test,DC=ps-scripting,DC=com
Properties    : {usncreated, objectclass, distinguishedname,objectguid...}

Path          : LDAP://ps-scripting.com/OU=HR,DC=ps-scripting,DC=com
Properties    : {usncreated, objectclass, distinguishedname,objectguid...}

Path          : LDAP://ps-scripting.com/OU=SubOU1,OU=Test,
                DC=ps-scripting,DC=com
Properties    : {usncreated, objectclass, distinguishedname,objectguid...}
Path          : LDAP://ps-scripting.com/OU=SubOU2,OU=Test,
                DC=ps-scripting,DC=com
Properties    : {usncreated, objectclass, distinguishedname,objectguid...}
```

¿Desea contarlos? Es muy fácil.

```
PS > $objBusqueda.FindAll().count
5
```

Ahora supongamos que sólo queremos encontrar las OU que comienzan por la letra «T». Para ello, vamos a tener que aportar una modificación a la propiedad `Filter` dotándola de otro criterio; como mostramos a continuación:

```
PS > $objBusqueda.Filter='(&(objectCategory=organizationalUnit)(ou=t*))'
PS > $objBusqueda.FindAll()
```

## b. Obtener la lista de los usuarios

La extracción de la lista de usuarios se hace exactamente de la misma manera que para los UO, con la diferencia del filtro. En efecto, sólo la clase de objeto difiere.

Liste todos los usuarios del AD DS a partir de la raíz:

```
PS > $objDominio = [ADSI]''
PS > $objBusqueda =
    New-Object System.DirectoryServices.DirectorySearcher($objDominio)
PS > $objBusqueda.Filter='(&(objectCategory=person)(objectClass=user))'
```

```
PS > $objBusqueda.FindAll()
```

Path	Properties
-----	-----
LDAP://CN=Administrador,CN=Users,DC=ps-scriptin...	{samaccounttype, lastlogon, lastlogontimestamp, objectsi...}
LDAP://CN=Invitado,CN=Users,DC=ps-scripting,DC=com	{samaccounttype, lastlogon, objectsid, whencreated...}
LDAP://CN=SUPPORT_388945a0,CN=Users,DC=ps-script...	{samaccounttype, lastlogon, objectsid, whencreated...}
LDAP://CN=oscar,OU=HR,DC=ps-scripting,DC=com	{samaccounttype, countrycode, cn, lastlogoff...}
LDAP://CN=krbtgt,CN=Users,DC=ps-scripting,DC=com	{samaccounttype, lastlogon, objectsid, whencreated...}
LDAP://CN=MyerKen,OU=Test,DC=ps-scripting,DC=com	{lastlogon, objectsid, whencreated, badpasswordtime...}
LDAP://CN=Usuario tititoto,OU=Test,DC=ps-scr...	{lastlogon, objectsid, whencreated, primarygroupid...}
LDAP://CN=Usuario tititoto2,OU=Test,DC=ps-sc...	{lastlogon, objectsid, whencreated, badpasswordtime...}



Para descubrir en detalle cómo realizar filtros potentes, le recomendamos encarecidamente consultar las páginas siguientes del sitio MSDN: <http://msdn2.microsoft.com/en-us/library/ms808539.aspx> (Creating More Efficient...) <http://msdn2.microsoft.com/en-us/library/aa746475.aspx> (Search Filter Syntax)

Si no se encuentra muy cómodo con la definición de filtros, le aconsejamos utilizar la herramienta de administración «Usuarios y equipos de Active Directory». Así, gracias a ella va a poder crear gráficamente, en algunos clics, las consultas personalizadas, probarlas y recuperar el valor del campo «Cadena de búsqueda». Este campo corresponde en realidad a la propiedad `Filter` del objeto `DirectorySearcher`. Sólo nos queda copiar/pegar este valor y nuestra consulta estará creada.

Creación de un filtro de búsqueda LDAP con las herramientas Windows

### c. Obtener la lista de los grupos

Siempre basándonos en el mismo principio, vamos a poder recuperar la lista de los grupos. De hecho toda la dificultad reside en la definición del filtro. Se trata de una dificultad relativa en la medida en que lo más «complicado» es finalmente conocer las numerosas clases y los atributos que componen el esquema de Active Directory.



Para descubrir los atributos y clases que componen el esquema así como sus valores, le recomendamos la herramienta de Support Tools llamada ADSIEdit.msc. ¡Pero tenga cuidado de no modificar nada si no está seguro!

Para listar los grupos vamos a efectuar un filtro en todos los objetos cuyo atributo `objectCategory` sea un grupo. Nuestro filtro será por tanto de la siguiente forma:

```
PS > $objDominio = [ADSI]''
PS > $objBusqueda = New-Object `
System.DirectoryServices.DirectorySearcher($objDominio)
PS > $objBusqueda.Filter='(objectCategory=group)'
```

Estas instrucciones nos devuelven todos los grupos, sin distinciones, contenidos en nuestro Active Directory.

Como usted sabrá, existen diversas clases de grupos: los locales, los globales y los universales. Para recuperar únicamente unos u otros, vamos a tener que perfeccionar nuestro filtro.

Existe un atributo que especifica justamente cual es el tipo de grupo; se trata del atributo `GroupType`. Éste puede contener un valor combinación de los valores siguientes:

Valor	Descripción
2	Grupo global.
4	Grupo local.
8	Grupo universal.
2147483648 (0x80000000)	Grupo de seguridad. Si este valor no está especificado, es que se trata de un grupo de distribución.

En resumen, si buscamos todos los objetos grupos cuyo atributo `GroupType` tenga el valor 0x80000002 (en hexadecimal) o 2147483650 (en decimal), encontraremos todos los grupos globales de seguridad. Y es precisamente lo que vamos a hacer:

```
PS > $objDominio = [ADSI]''
PS > $objBusqueda = New-Object `
    System.DirectoryServices.DirectorySearcher($objDominio)
PS > $objBusqueda.Filter='(&(objectCategory=group)(GroupType= 2147483650))'
```

```
PS > $objBusqueda.FindAll()

Path                                     Properties
----
LDAP://ps-scripting.com/CN=Ordenadores del dominio... {objectguid,
groupType, obj...}
LDAP://ps-scripting.com/CN=Controladores del dominio... {objectguid,
groupType, obj...}
LDAP://ps-scripting.com/CN=Admins del dominio,CN=... {samaccounttype,
objectguid...}
LDAP://ps-scripting.com/CN=Usuario del dominio,C... {objectguid,
```

```

grouptype, obj...}
LDAP://ps-scripting.com/CN=Invitados del dominio,CN... {objectguid,
grouptype, obj...}
LDAP://ps-scripting.com/CN=Propietarios creador... {objectguid,
grouptype, obj...}
LDAP://ps-scripting.com/CN=DnsUpdateProxy,CN=Use... {objectguid,
grouptype, obj...}
LDAP://ps-scripting.com/CN=MiGrupo,OU=Test,DC=... {objectguid,
grouptype, obj...}
LDAP://ps-scripting.com/CN=test,OU=HR,DC=powersh... {usncreated,
cn, grouptype,...}

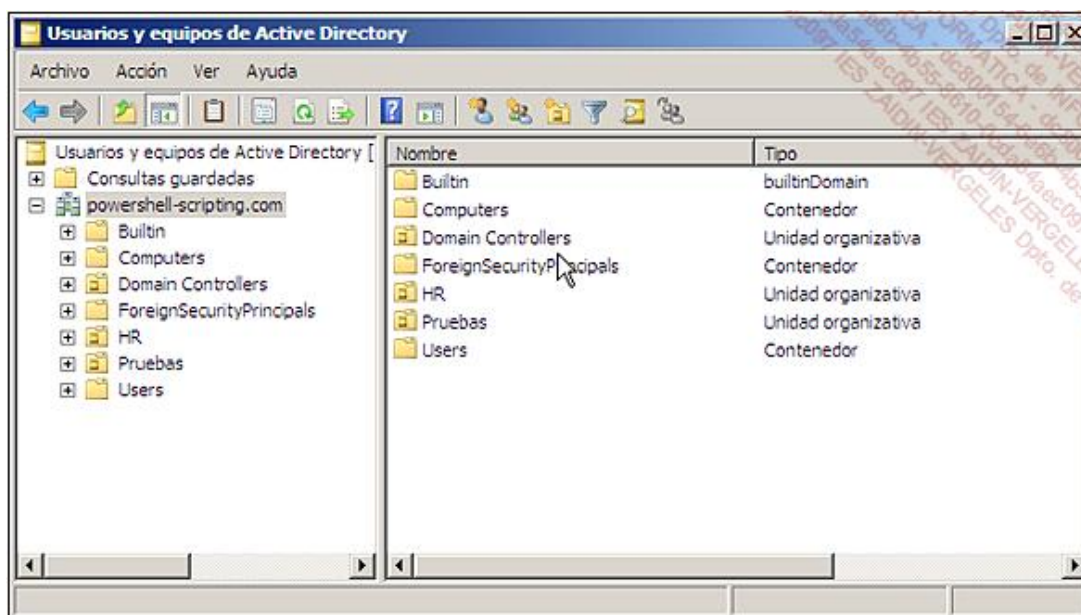
```

### 3. Administración de las unidades organizativas (UO)

Para aquellos que no lo conozcan, una unidad organizativa es un contenedor en el que vamos a poder clasificar los objetos. Es interesante crear UO cuando hay muchos objetos por clasificar, y lo es por cuestiones de organización. En efecto, cuando hay varios miles de cuentas de usuarios y grupos en una misma OU es muy fácil perderse. Además, el otro punto interesante de las UO es que es posible aplicar (o dicho de otro modo «vincular») políticas de grupos o GPO (*Group Policy Object*).

Una política de grupo es un conjunto de parámetros definidos de forma centralizada por el administrador del dominio. Estos parámetros, que corresponden generalmente a las claves de registro, se aplican a los objetos contenidos en el interior de la UO; estos objetos son en su mayoría de tipo usuario u equipo.

Veamos aquí un árbol de dominio típico:



Árbol de dominio visto con «Usuarios y equipos de Active Directory»

Podrá constatar que determinadas UO poseen un icono que representa una carpeta con un pequeño libro en el interior. Esto significa que estas unidades organizativas pueden estar relacionadas con las políticas de grupos. Hay que saber que las UO denominadas «built-in», es decir, creadas por defecto en el sistema, no pueden asociarse a GPO; excepto de la UO «Domain Controllers».

Otra ventaja de crear una UO, es que es posible delegar su gestión a un usuario que no sea miembro del grupo «Admins del dominio». Esto puede permitir, por ejemplo, a un administrador de UO controlar los miembros de los grupos (a condición de que los grupos y los usuarios que controla sean miembros de su UO). Se podría, además, y si el administrador le ha otorgado los permisos correspondientes, reiniciar las contraseñas de sus colaboradores.



Finalmente, tenga en cuenta que es posible anidar varias unidades organizativas.

## a. Crear una unidad organizativa

Como siempre, todo empieza por la conexión al servicio de directorio. Debemos conectarnos al lugar donde queremos crear la UO. Por ejemplo, creamos una UO en la raíz de AD DS:

```
PS > $objDominio = [ADSI]''
PS > $objOU = $objDominio.Create('organizationalUnit','ou=Finanzas')
PS > $objOU.Put('description', 'Servicios financieros')
PS > $objOU.SetInfo()
```

Acabamos de crear la UO denominada «Finanzas» y le hemos asignado una descripción. Ahora, crearemos una nueva UO en el interior de ésta que contendrá los objetos relativos a la contabilidad:

```
PS > $adsPath = 'LDAP://OU=Finanzas,' + ([ADSI]'').distinguishedName
PS > $objDominio = [ADSI]$adsPath
PS > $objOU = $objDominio.Create('organizationalUnit','ou=Contabilidad')
PS > $objOU.Put('description','Departamento contable')
PS > $objOU.SetInfo()
```

## b. Renombrar una unidad organizativa

Hay una reestructuración en su empresa que hace que el Servicio financiero se haya ampliado, y englobado el servicio de control de gestión. Como buen administrador de sistema, renombrará la UO de «finanzas» a «finanzas y gestión».

Para ello utilizará las siguientes líneas de código:

```
PS > $objOU = [ADSI]''
PS > $objOU.MoveHere('LDAP://OU=Finanzas,DC=ps-scripting,DC=com',
'OU=Finanzas y gestion')
```

Renombrar objetos en Active Directory se efectúa gracias al método `MoveHere`. El cual puede utilizarse también para el movimiento de objetos.

## c. Movimiento de objetos en una unidad organizativa

### Movimiento de un grupo

Una vez más el método `MoveHere` va a sernos útil. En este ejemplo vamos a mover el grupo «Encargos de clientes» de la UO «Informática» a la UO «Usuarios».

Para ello, será necesario conectarse a la UO de destino ya que el método `MoveHere` pertenece a esta última. A continuación tenemos que decirle a `MoveHere` el DN del grupo a mover, al igual que su nuevo nombre (nombre que puede ser idéntico a su nombre original, sino desea modificarlo).

```
PS > $objUODest = [ADSI]'LDAP://OU=Usuarios,DC=ps-scripting,DC=com'
PS > $objUODest.MoveHere('LDAP://CN=Encargos de clientes,OU=Informática,
DC=ps-scripting,DC=com', 'CN=Encargos de clientes')
```

Observe que empezamos por especificar la UO de destino en lugar de la UO origen como sería habitual. Aunque esto pueda parecer algo confuso, debe razonar al revés planteándose la siguiente pregunta: «¿A donde debe ir mi objeto?».

Finalmente, en el segundo argumento de `MoveHere`, podrá especificar el nuevo nombre del objeto si desea renombrarlo al mismo tiempo que lo desplaza.

### **Movimiento de un usuario**

En este ejemplo, moveremos el usuario Oscar, de la UO «Usuarios» a la UO «Users».

```
PS > $objUODest = [ADSI]'LDAP://CN=Users,DC=ps-scripting,DC=com'
PS > $objUODest.MoveHere('LDAP://CN=Oscar,OU=Usuarios,DC=ps-scripting,DC=com', 'CN=Oscar')
```

### **Movimiento de una UO**

Para ilustrar el movimiento de una unidad organizativa, moveremos la UO llamada «Usuarios» situada en el raíz de AD DS a la UO «Informática».

```
PS > $objUODest = [ADSI]'LDAP://OU=Informática,DC=ps-scripting,DC=com'
PS > $objUODest.MoveHere('LDAP://OU=Usuarios,DC=ps-scripting,DC=com', 'OU=Usuarios')
```

Observe que podemos de igual forma renombrar la UO que hemos desplazado especificando el nuevo nombre como segundo argumento en el método `MoveHere`.

## **d. Eliminar una unidad organizativa**

La eliminación de una UO se realiza en dos etapas:

- Conexión al contenedor en el que se encuentra la UO.
- Eliminación de la UO.

Supongamos esta vez que nuestra empresa ha externalizado la gestión y las finanzas. En estas condiciones, la UO «Finanzas y gestión» ya no será necesaria. Por este motivo vamos a eliminarla, con el script siguiente:

```
PS > $objOU = [ADSI]''
PS > $objOU.Delete('organizationalUnit','OU=Finanzas y gestion')
```

Para que el script funcione, será necesario que la UO esté completamente vacía.

## **4. Administración de los grupos**

### **a. Crear un grupo**

Antes de crear un grupo, usted necesita conocer su ámbito. Puede ser global, local o universal. Si no se especifica el ámbito, entonces por defecto, se creará un grupo global. Para nuestros ejemplos, hemos creado una UO llamada «Informática» y crearemos grupos en su interior.



Para más información sobre los grupos de seguridad, sus roles y su extensión, puede visitar el sitio Microsoft TechNet, en la dirección siguiente: [http://technet.microsoft.com/es-es/library/cc781446\(Ws.10\).aspx](http://technet.microsoft.com/es-es/library/cc781446(Ws.10).aspx)

### **Creación de un grupo global**

Creemos un grupo llamado «Responsables proyectos». Tenga en cuenta que si no informa el `SamAccountName`, Windows asignará uno por defecto, el cual no tendrá un nombre tan inteligible. Puede especificar cualquier valor pero le aconsejamos poner el mismo que el CN.

```
PS > $objOU = [ADSI]'LDAP://OU=Informática, DC=ps-scripting,DC=com'
```

```
PS > $objGrupo = $objOU.Create('group', 'cn=Responsables proyectos')
PS > $objGrupo.Put('samaccountname', 'Responsables proyectos')
PS > $objGrupo.Put('description','Responsables de los proyectos informáticos')
PS > $objGrupo.SetInfo()
```

### Creación de un grupo local

Para crear un grupo de un ámbito distinto a un grupo global tendremos que especificar un valor para el atributo `GroupType`. La lista de los valores es la misma que la que hemos utilizado para efectuar un filtro de búsqueda.

Para un grupo local, el valor de `GroupType` será **2147483652**.

Creamos un grupo local, siempre en la UO «Informática», llamado «Encargos de clientes», como se muestra a continuación:

```
PS > $objOU = [ADSI]'LDAP://OU=Informática, DC=ps-scripting,DC=com'
PS > $objGrupo = $objOU.Create('group', 'cn=Encargos de clientes')
PS > $objGrupo.Put('samaccountname', 'Encargos de clientes')
PS > $objGrupo.Put('groupType', '2147483652')
PS > $objGrupo.Put('description', 'Encargos de clientes informática')
PS > $objGrupo.SetInfo()
```

### Creación de un grupo universal

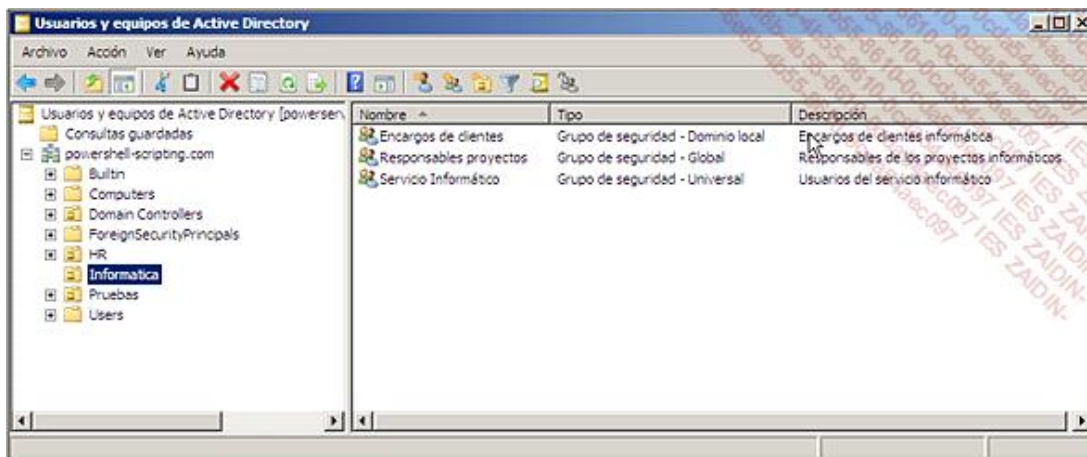
Nada más fácil que crear un grupo universal de seguridad; bastará con modificar nuevamente el valor del atributo `GroupType`.

Para un grupo local, el valor de `GroupType` será **2147483656**.

Creamos el grupo universal llamado «Servicio Informático», como se muestra a continuación:

```
PS > $objOU = [ADSI]'LDAP://OU=Informática,DC=ps-scripting, DC=com'
PS > $objGrupo = $objOU.Create('group', 'cn=Servicio Informático')
PS > $objGrupo.Put('samaccountname', 'Servicio Informático')
PS > $objGrupo.Put('groupType', '2147483656')
PS > $objGrupo.Put('description', 'Usuarios del Servicio Informático')
PS > $objGrupo.SetInfo()
```

Verificamos la creación de todos nuestros grupos:



Verificación de la creación de los grupos

## b. Asignar uno o varios miembros a un grupo

### Adición de un miembro

Para añadir un objeto a un grupo, basta con conectarse al grupo y luego utilizar el método `Add`. Este método conserva el contenido existente del grupo y añade el objeto especificado como argumento. El objeto puede ser un usuario u otro grupo si se quiere hacer una anidación de grupos. Por último para validar la modificación, no debe olvidarse de utilizar el método `SetInfo`.

Supongamos que queremos añadir el usuario «Javier» al grupo «Encargos de clientes»; este último situado en la UO «Informática».

```
PS > $objGrupo = [ADSI]'LDAP://CN=Encargos de clientes,
OU=Informática,DC=ps-scripting,DC=com'
PS > $objGrupo.Add('LDAP://CN=Javier,OU=Usuarios,
DC=powershell-scripting,DC=com')
PS > $objGrupo.SetInfo()
```

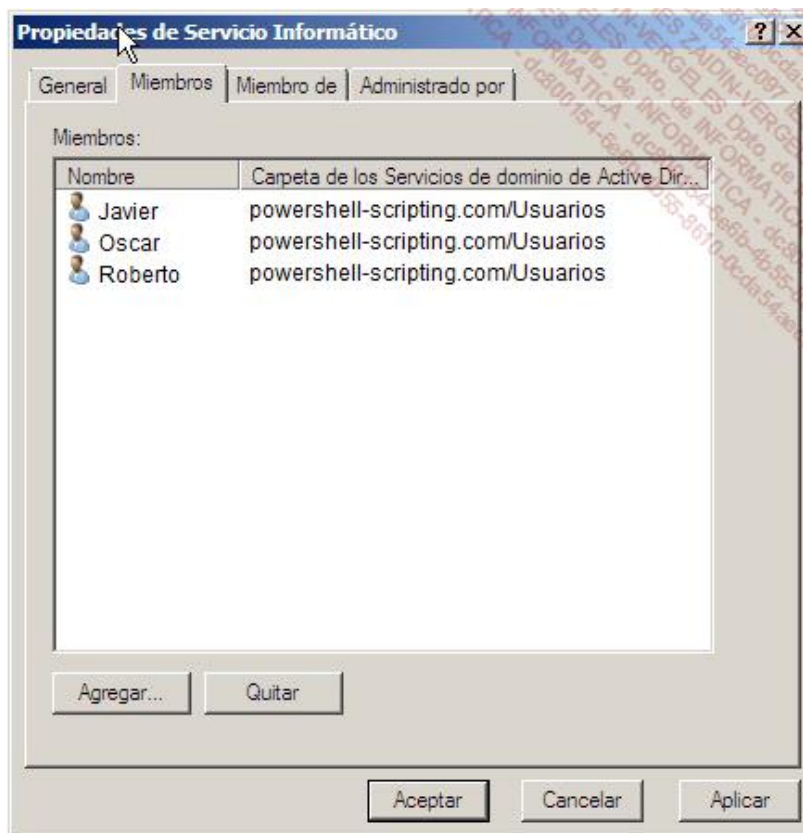
### Adición de varios miembros

Es muy fácil añadir varios miembros a un grupo. Bastará con utilizar el método `Add` como para añadir un usuario, y esto tantas veces como objetos a añadir al grupo.

Por ejemplo, para añadir los tres usuarios Javier, Roberto y Oscar al grupo «Servicio Informático», utilizamos el código siguiente:

```
PS > $objGrupo = [ADSI]'LDAP://CN=Servicio Informático,OU=Informática,
DC=ps-scripting,DC=com'
PS > $objGrupo.Add('LDAP://CN=Javier,OU=Usuarios,
DC=powershell-scripting,DC=com')
PS > $objGrupo.Add('LDAP://CN=Roberto,OU=Usuarios,
DC=powershell-scripting,DC=com')
PS > $objGrupo.Add('LDAP://CN=Oscar,OU=Usuarios,
DC=powershell-scripting,DC=com')
PS > $objGrupo.SetInfo()
```

Verificamos el resultado con la consola gráfica:



Verificación de los miembros de un grupo

### c. Renombrar un grupo

Al igual que para renombrar una unidad organizativa, haremos una llamada al método `MoveHere`.

En el ejemplo siguiente, renombraremos el grupo «Servicio Informático» como «Servicio Informático de gestión»:

```
PS > $objGrupo = [ADSI]'LDAP://OU=Informática,DC=ps-scripting,DC=com'
PS > $objGrupo.MoveHere('LDAP://CN=Servicio Informático,OU=Informática,
DC=ps-scripting,DC=com', 'CN=Servicio Informático de gestión')
```

La primera línea nos conecta a la UO que contiene el grupo. La segunda hace la llamada al método `MoveHere` donde indicamos en el primer argumento el Distinguished Name del grupo (el nombre completo LDAP), después el nuevo nombre en formato abreviado «CN=Nuevo nombre». Esta nomenclatura se denomina RDN por *Relative Distinguished Name*.



Para mover un grupo, consulte la sección [Movimiento de objetos en una unidad organizativa](#).

### d. Eliminar un grupo

Para eliminar un grupo, utilizaremos el método `Delete`. Es posible eliminar un grupo sin necesidad de eliminar previamente su contenido. Sepa también que al eliminar un grupo no se eliminan los objetos contenidos en su interior.

Eliminaremos, por ejemplo, el grupo «Encargos de clientes» situado en la UO «Usuarios»:

```
PS > $objGrupo = [ADSI]'LDAP://OU=Usuarios,DC=ps-scripting,DC=com'
PS > $objGrupo.Delete('group', 'CN=Encargos de clientes')
```

## 5. Administración de los usuarios

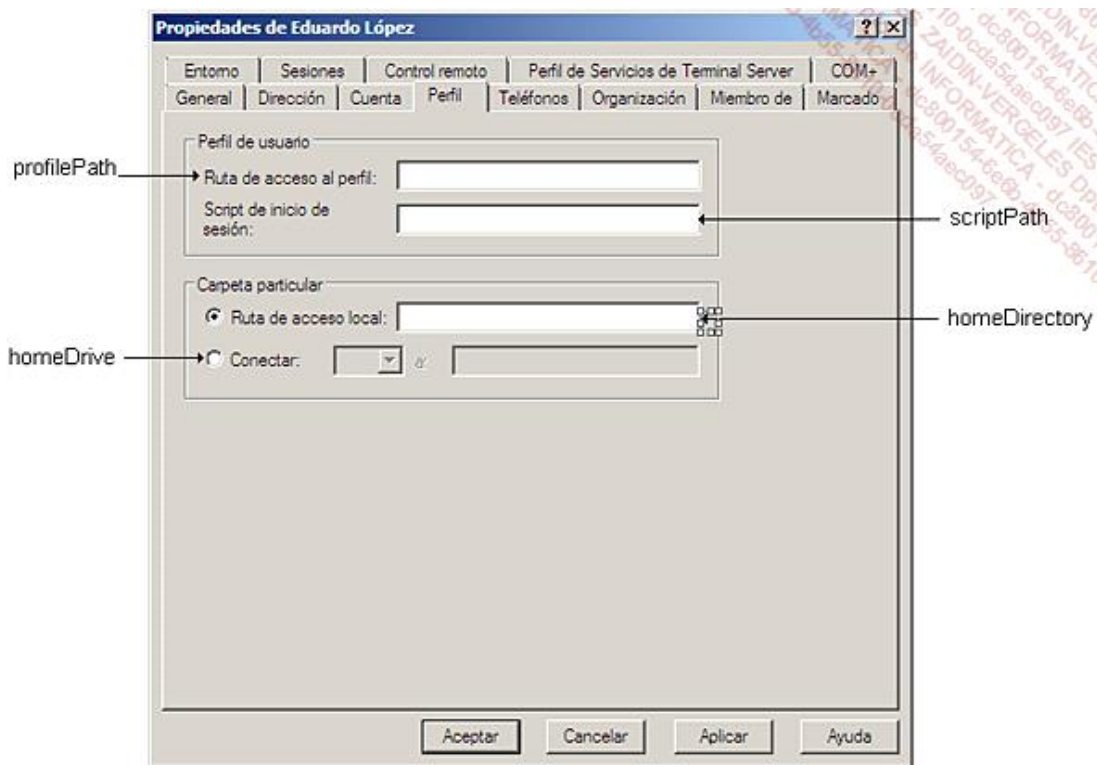
### a. Crear una cuenta de usuario

La creación de las cuentas de usuario forma parte del trabajo diario de los administradores de sistemas. Veremos en el próximo capítulo cómo automatizar esta tarea a menudo tediosa y repetitiva. Mientras tanto, veremos los principales atributos a definir cuando se crea una cuenta.

Con el fin de visualizar mejor estos atributos y descubrir sus «verdaderos» nombres, entendemos que corresponden a sus nombres definidos en el Esquema Active Directory, vamos a descubrir los diferentes campos de un usuario tal y como se muestran cuando utilizamos la consola **Usuarios y Equipos Active Directory**.

Propiedades de una cuenta de usuario - pestaña **General**

Propiedades de una cuenta de usuario - pestaña **Cuenta**



Propiedades de una cuenta de usuario - pestaña **Perfil**


Para crear un usuario, será necesario informar como mínimo los atributos siguientes:

- cn,
- sAMAccountName.

#### Ejemplo 1:

Creación del usuario «Eduardo López» en la UO «Usuarios».

```
PS > $objOU=[ADSI]'LDAP://OU=Usuarios,DC=ps-scripting,DC=com'
PS > $objUser=$objOU.Create('user', 'cn=Eduardo López')
PS > $objUser.Put('SamAccountName', 'López')
PS > $objUser.SetInfo()
```

 Aunque estos dos atributos son suficientes para crear un usuario, le recomendamos encarecidamente añadir el atributo `UserPrincipalName`. Este atributo llegó con Windows 2000/ Active Directory; permite «ganar en seguridad» ya que en la autenticación de un usuario activa la autenticación Kerberos en lugar de NTLM.

#### Ejemplo 2:

Creación de un usuario con `UserPrincipalName`.

```
PS > $objOU=[ADSI]'LDAP://OU=Usuarios,DC=ps-scripting,DC=com'
PS > $objUser=$objOU.Create('user', 'cn=Eduardo López')
PS > $objUser.Put('SamAccountName', 'López')
PS > $objUser.Put('UserPrincipalName', 'lopez@powershell-scripting.com')
PS > $objUser.SetInfo()
```



## b. Asignación de una contraseña

Acabamos de crear un usuario pero no le hemos definido todavía una contraseña. Para ello, y como para la gestión de los usuarios locales, vamos a usar la propiedad `SetPassword`.



Al contrario que los usuarios locales, la definición de una contraseña no es obligatoria para la creación de una cuenta en el servicio de directorio Active Directory. Si se quiere que el usuario pueda conectarse con su cuenta, esta etapa es ineludible.

Tenga en cuenta también que para asignar una contraseña a un usuario, es necesario que este último ya esté creado. En otras palabras, no puede de una sola vez crear un usuario con sus diferentes atributos, y asignarle una contraseña. Esto no supone un gran inconveniente, ipero hay que saberlo!

Estará por tanto obligado a utilizar `SetInfo` una primera vez para la creación del usuario, y después una segunda vez, para validar la definición de su contraseña.

### Ejemplo 1:

*Asignamos una contraseña a un usuario ya existente.*

```
PS > $objUser=[ADSI]'LDAP://CN=Eduardo López,OU=Usuarios,DC=ps-scripting,DC=com'
PS > $objUser.SetPassword('P@ssw0rd')
PS > $objUser.SetInfo()
```



Asegúrese de introducir una contraseña que corresponda a las reglas de seguridad que habrá definido en Active Directory Domain Services, sino obtendrá un mensaje de error cuando intente de asignar la contraseña.

### Ejemplo 2:

*Creación de un usuario y asignación de una contraseña.*

```
PS > $objOU=[ADSI]'LDAP://OU=Usuarios,DC=ps-scripting,DC=com'
PS > $objUser=$objOU.Create('user', 'cn=Eduardo López')
PS > $objUser.Put('SamAccountName', 'López')
PS > $objUser.Put('UserPrincipalName', 'lopez@powershell-scripting.com')
PS > $objUser.SetInfo()
PS > $objUser.SetPassword('P@ssw0rd')
PS > $objUser.SetInfo()
```

## c. Activación de una cuenta de usuario

Cuando creamos las cuentas de usuarios con ADSI en el AD DS, éstas se crean desactivadas. Debemos por tanto activarlas para que los usuarios puedan hacer uso de sus cuentas.

La activación de una cuenta del AD DS se realiza como para una cuenta local. Podemos por tanto utilizar dos técnicas: la primera consiste en modificar la propiedad `AccountDisabled`, y la segunda consiste en modificar el valor del atributo `UserAccountControl`.

### Primera técnica:

*Modificación de una cuenta con la propiedad `AccountDisabled`.*



```
PS > $objUser=
[ADSI]'LDAP://CN=Eduardo López,OU=Usuarios,DC=ps-scripting,DC=com'
PS > $objUser.PSBase.InvokeSet('AccountDisabled', $False)
PS > $objUser.SetInfo()
```

Para verificar por script que la propiedad `AccountDisabled` se ha modificado correctamente, haremos lo siguiente:

```
PS > $objUser=
[ADSI]'LDAP://CN=Eduardo López,OU=Usuarios,DC=ps-scripting,DC=com'
PS > $objUser.PSBase.InvokeGet('AccountDisabled')

False
```

Obtenemos el valor `False`, lo que significa que la cuenta no está desactivada; el usuario está por tanto activo.

#### Segunda técnica:

##### *Modificación de la propiedad `UserAccountControl`.*

Esta técnica es muy interesante, ya que nos permite hacer muchas más cosas que simplemente activar o desactivar una cuenta. Esta propiedad define también, entre otras cosas, si un usuario debe cambiar su contraseña en la próxima conexión o si no puede cambiarla, etc.

Para mayor información acerca de lo que se puede hacer con esta propiedad, consulte la URL siguiente: <http://support.microsoft.com/kb/305144>.

Para activar una cuenta, es preciso realizar un **Y** lógico en el complemento de **2** sobre el valor actual de la propiedad `UserAccountControl`, como se indica a continuación:

```
PS > $objUser=
[ADSI]'LDAP://CN=Eduardo López,OU=Usuarios,DC=ps-scripting,DC=com'
PS > $objUser.userAccountControl[0] =
$objUser.userAccountControl[0] -band (-bnot 2)
PS > $objUser.SetInfo()
```

## **d. Lectura/definición de atributos**

La primera cosa que debemos hacer para leer o definir los atributos de un usuario consiste en conectarnos a este último por medio de una consulta ADSI.

Imaginemos que queremos definir el atributo `Description` del usuario Eduardo López, creado en el ejemplo anterior:

```
PS > $objUser=
[ADSI]'LDAP://CN=Eduardo López,OU=Usuarios,DC=ps-scripting,DC=com'
PS > $objUser.Put('Description', 'Cuenta de usuario')
PS > $objUser.SetInfo()
```

Ahora, vamos a leer el valor de la propiedad para verificar que ésta ha sido tomada en cuenta correctamente por el sistema:

```
PS > $objUser=
[ADSI]'LDAP://CN=Eduardo López,OU=Usuarios,DC=ps-scripting,DC=com'
PS > $objUser.Get('Description')

Cuenta de Usuario
```

También podríamos haber hecho lo siguiente:

```
PS > $objUser.Description
```

Cuenta de Usuario

## e. Eliminación de atributos

Es tentador pensar que se puede asignar a un atributo una cadena vacía o una cadena con un espacio para borrar el valor que contiene. Pero la realidad es otra: la asignación de una cadena vacía no está autorizada, y la asignación de una cadena con un espacio no elimina el atributo sino que asignará una nueva cadena. Para que se convenza bastará con iniciar ADSIEdit y posicionarse en el usuario a manipular. De este modo podrá ver fácilmente los atributos que tienen un valor y los que no tienen ninguno.

Resumiendo, en otras palabras, no se empleará para esa tarea el método `Put` sino el método `PutEx`.

Este método se utiliza habitualmente con los atributos multivalor. Puede ser una alternativa al método `Add` que hemos utilizado para añadir miembros a un grupo.

El método `PutEx` tiene tres argumentos. El primero indica lo que se desea hacer:

Valor	Operación
1	<b>elimina</b> el contenido completo del atributo.
2	<b>reemplaza</b> el o los valores actuales por el o los valores especificados.
3	<b>añade</b> los valores al valor actual.
4	<b>borra</b> únicamente el valor especificado.

El segundo argumento es el nombre del atributo que realiza la operación. Por último, el tercer argumento es el o los valores a definir.

En nuestro caso (la eliminación), utilizaremos el valor 1 como primer argumento. Supongamos ahora que queremos eliminar la descripción del usuario «Eduardo López».

```
PS > $objUser=[ADSI]'LDAP://CN=Eduardo López,OU=Usuarios,DC=ps-scripting,DC=com'
PS > $objUser.PutEx(1, 'Description', $null)
PS > $objUser.SetInfo()
```



Tanto si desea utilizar el método `PutEx` para actualizar los atributos multivalor o no, deberá pasar siempre como valor un objeto de tipo tabla. Por ejemplo, si queremos definir el atributo `Teléfono` de Eduardo López debemos pasar el número de teléfono en formato de tabla conteniendo un sólo elemento:

```
PS > $objUser=[ADSI]'LDAP://CN=Eduardo López,OU=Usuarios,DC=ps-scripting,DC=com'
PS > $objUser.PutEx(2, 'telephoneNumber', @('556102030'))
PS > $objUser.SetInfo()
```

Y tanto para especificar una lista de valores, como para definir otros números de teléfono, utilizaremos siempre una tabla de valores, como en este ejemplo:

```
PS > $objUser=[ADSI]'LDAP://CN=Eduardo López,OU=Usuarios,DC=ps-scripting,DC=com'
PS > $objUser.PutEx(2, 'otherTelephone', @('123', '456', '789'))
```

```
PS > $objUser.SetInfo()
```

Seguramente habrá observado que existía otra forma de eliminar un valor de los atributos. Ésta consiste en eliminar un valor entre un conjunto de valores que figuran en un atributo multivalor tal como `otherTelephone` (véase el caso anterior).

Si queremos suprimir únicamente el valor «456» debemos escribir lo siguiente:

```
PS > $objUser=[ADSI]'LDAP://CN=Eduardo López,OU=Usuarios,DC=ps-scripting,DC=com'
PS > $objUser.PutEx(4, 'otherTelephone', @('456'))
PS > $objUser.SetInfo()
```

## f. Eliminar un usuario

La eliminación de usuarios es fácil pero tenga en cuenta que es una actividad peligrosa ya que no tiene vuelta atrás. Por esta razón, es aconsejable antes de eliminar una cuenta, desactivarla algún tiempo antes.

Para eliminar una cuenta, utilizaremos el método `Delete`. Le asociaremos dos argumentos: en primer lugar la clase de objeto a eliminar, en nuestro caso «user» y en segundo lugar, el RDN del objeto a eliminar en formato «CN=nombre».

En cambio, esta vez no nos conectaremos al usuario propiamente dicho, sino a la unidad organizativa que contiene el usuario.

Eliminaremos por ejemplo la cuenta de Eduardo López:

```
PS > $objOU=[ADSI]'LDAP://OU=Usuarios,DC=ps-scripting,DC=com'
PS > $objOU.Delete('user', 'CN=Eduardo López')
```

Si por algún motivo no conocemos la ubicación del usuario que queremos eliminar, podemos hacer una llamada para localizarlo a `DirectorySearcher`. Una vez localizado el objeto, nos conectaremos a la UO que lo contiene, después pasaremos su RDN al método `Delete`.

```
PS > $UserCN = 'Juan Carlos Durante'
PS > $objDominio = [ADSI]''
PS > $objBusqueda = New-Object `
System.DirectoryServices.DirectorySearcher($objDominio)
PS > $objBusqueda.Filter="(&(objectCategory=user)(cn=$UserCN))"
PS > $Busqueda=$objBusqueda.FindOne()
PS > $UO = $($($objBusqueda.FindOne().path) -replace "CN=$UserCN,", '')
PS > $objOU=[ADSI]"$UO" # Conexión a la UO
PS > $objOU.Delete('user',"CN=$UserCN") # Eliminar el usuario
```



Para asignar los grupos a un usuario, consulte la sección [Asignar uno o varios miembros a un grupo](#). Para mover un usuario en una UO, consulte la sección [Movimiento de objetos en una unidad organizativa](#).