El conjunto de comandos del módulo Active Directory

Antes de entrar de lleno en el tema, debemos hablar de la búsqueda de los objetos. En efecto, antes de actuar en un objeto, primero deberá llegar a identificarlo para conectarse y así poder aplicarle los comandos.

1. Búsqueda de objetos

La búsqueda de objetos con el conjunto de comandos Active Directory puede efectuarse al menos de tres formas diferentes, que son:

- búsqueda basada en la realización de un filtro LDAP (parámetro `LDAPFilter`).
- búsqueda basada en la realización de un filtro genérico (parámetro `Filter`).
- búsqueda basada en una identidad conocida con anterioridad (parámetro `Identity`).

La mayor parte de los commandlets AD poseen estos tres parámetros, es la razón por la cual es importante conocerlos mínimamente. Veremos que el hecho de disponer de estos modos de búsqueda ofrece una flexibilidad máxima.

a. Creación de un filtro LDAP

Si usted ya tiene consultas LDAP listas para utilizar o si conoce perfectamente la sintaxis LDAP, entonces el parámetro **`-LDAPFilter`** le será muy útil.

En los ejemplos que vamos a ver, centre principalmente su atención en el filtro más que en el commandlet utilizado.

Ejemplo:

Obtener los objetos donde la propiedad «name» contenga la cadena de caracteres «quez».

`-LDAPFilter '(name=*quez*)'`

```
PS > Get-ADObject -LDAPFilter '(name=*quez*)'

DistinguishedName : CN=Oscar Vázquez,CN=Users,DC=powershell-scripting,DC=com
Name              : Oscar Vázquez
ObjectClass       : user
ObjectGUID        : 7c67b7a3-2415-42c2-8e7a-77a435b3054c
```

Este ejemplo básico define un filtro en la propiedad «name». Asociado a un comando de tipo `Get-*`, este filtro nos devolverá todos los objetos que contienen «quez» en su nombre. El resultado será susceptible de contener todos los tipos de objetos posibles de Active Directory (user, computer, organizationalUnit, etc.) ya que no hemos precisado un tipo en particular.

Veamos ahora otro ejemplo:

`-LDAPFilter '(&(objectCategory=computer)(name=win*))'`

```
PS > Get-ADObject -LDAPFilter '(&(objectCategory=computer)(name=win*))'

DistinguishedName                                       Name
-----------------                                       ----
CN=WIN7_US_X64,CN=Computers,DC=powershell-scripting,DC=com  WIN7_US_X64
CN=WINXP,CN=Computers,DC=powershell-scripting,DC=com        WINXP
```

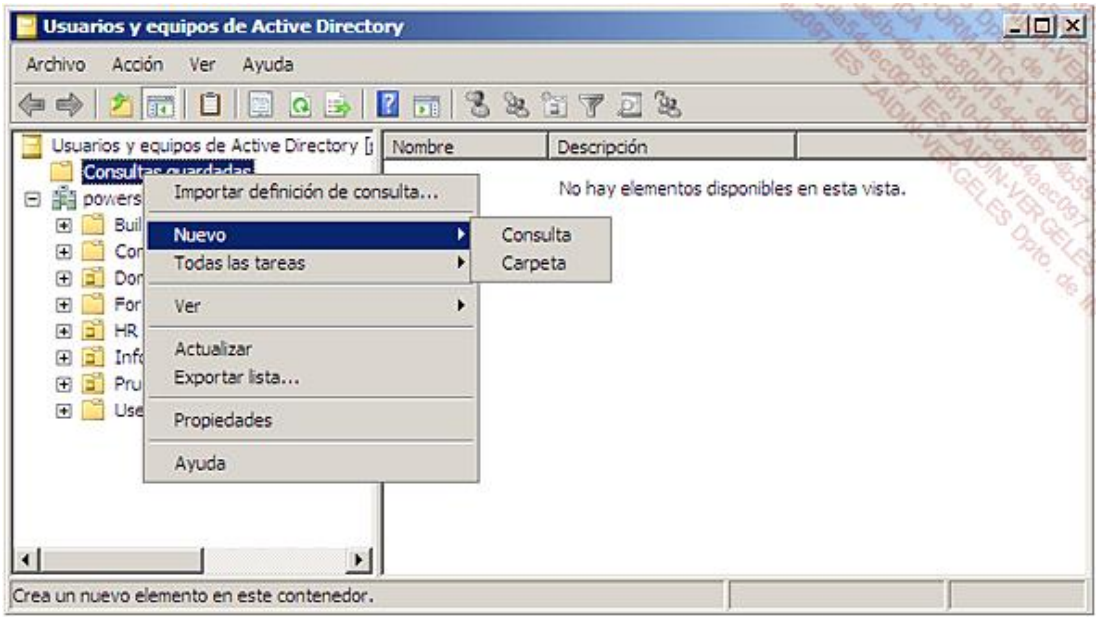
```
CN=WIN2K8X64,CN=Computers,DC=powershell-scripting,DC=com    WIN2K8X64
CN=WINXPPSV2,CN=Computers,DC=powershell-scripting,DC=com    WINXPPSV2
```

Este filtro nos devolverá todos los objetos de la categoría «computer» cuyo nombre empiece por la cadena de caracteres «win».

b. Creación avanzada de un filtro LDAP

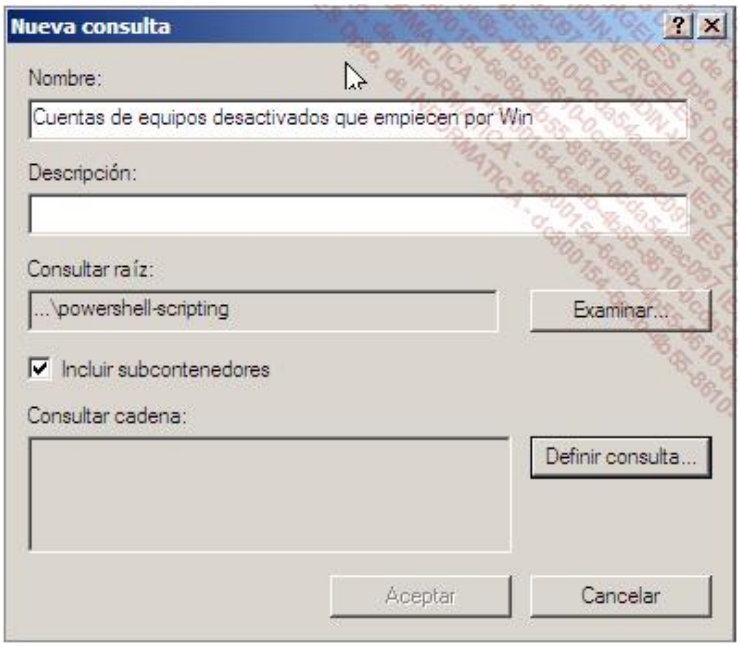
Lo que apreciamos especialmente con los filtros de tipo LDAP es que existe una fabulosa herramienta instalada en todos los controladores de dominio Windows que nos va a facilitar mucho la vida. Se trata de una funcionalidad bastante desconocida, pero útil, de la consola «Usuarios y equipos de Active Directory».

Esta consola trata de ser un enorme generador de peticiones LDAP. Es accesible desde la estructura «Consultas guardadas», veámoslo en la imagen:



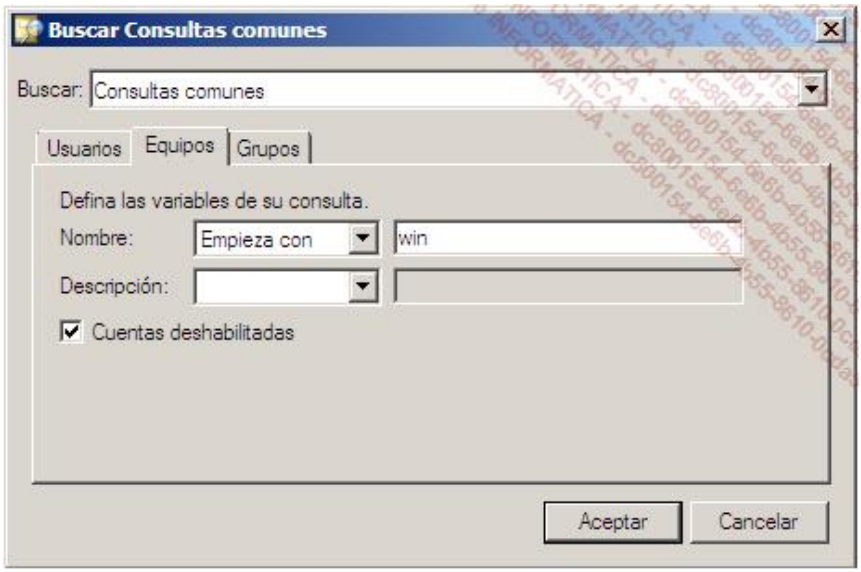
Editor de consultas LDAP en la consola de administración Active Directory

Para abrir el editor de consultas, pulse el botón secundario de su ratón en **Consultas guardadas** y luego en **Nuevo** - **Consulta**. Después asigne un nombre a su consulta, como se muestra a continuación:



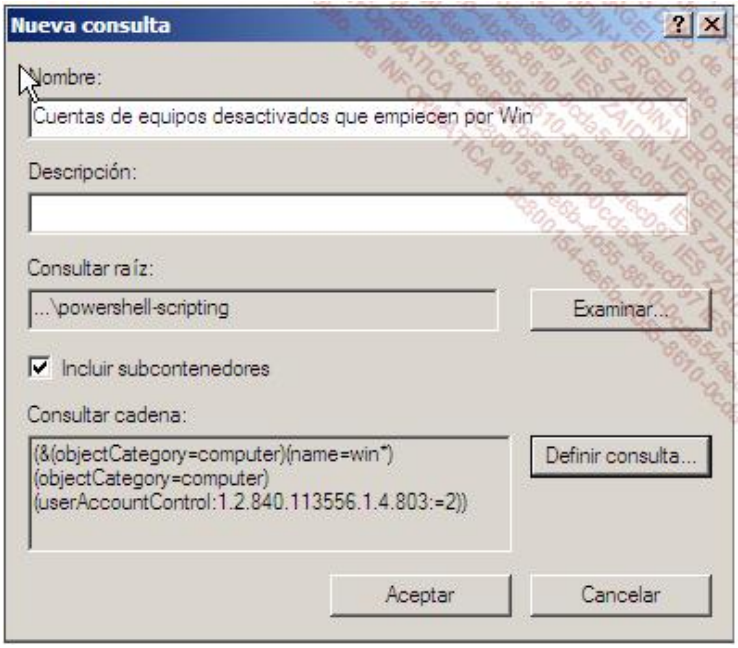
Editor de consultas LDAP - definición del nombre de la consulta

Seguidamente, pulse en el botón **Definir consulta…**. Aparecerá entonces una nueva ventana que incluye un cierto número de consultas predefinidas, y no tiene más que informar los campos que le interesan. En este ejemplo hemos escogido la pestaña **Equipos**, hemos completado el primer campo en blando con el inicio del nombre de la búsqueda, hemos seleccionado **Empieza con** y hemos marcado la opción **Cuentas deshabilitadas**.



Editor de consultas LDAP - definición de la consulta

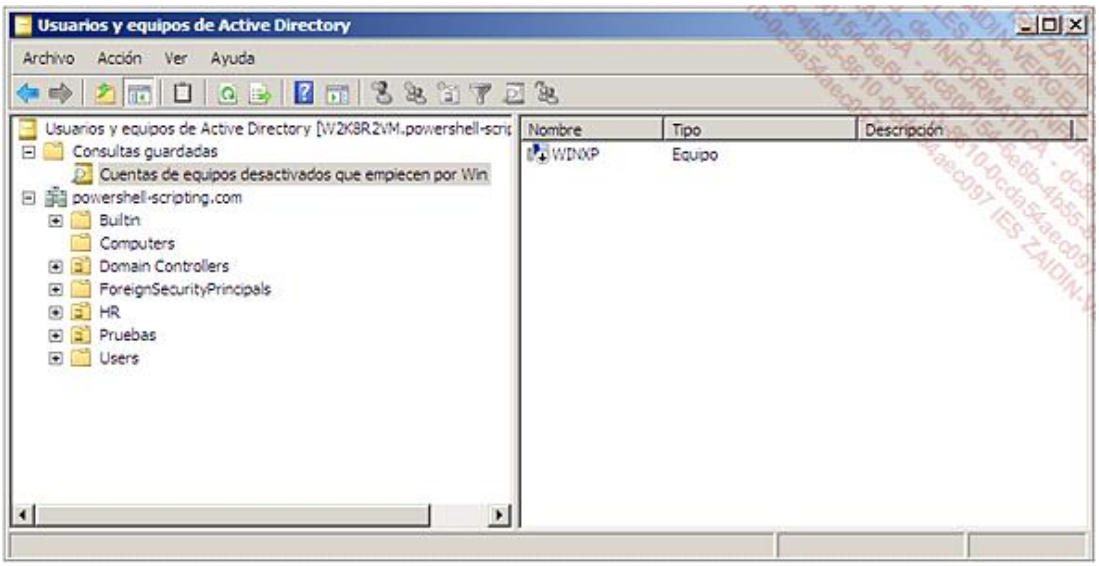
Valide la consulta pulsando en el botón **Aceptar**; esto nos lleva al menú principal. Ahora vamos a recoger el fruto de nuestro trabajo copiando en el portapapeles la petición LDAP presente en el campo **Consultar cadena**:



Editor de consultas LDAP - recuperación de la consulta

Y bien, hemos marcado algunas casillas, cumplimentado algunos campos y la consulta LDAP se ha generado automáticamente. ¿No es un instrumento fantástico?

Como guinda del pastel, cuando se cierra la ventana, volvemos a la interfaz principal de la consola **Usuarios y equipos de Active Directory** y podemos ver visualmente el resultado de nuestra consulta. Si ésta no se ajusta a nuestras expectativas, podemos modificarla y probarla hasta que el resultado nos convenga.



Visualización del resultado de la consulta definida gráficamente

No tenemos más que pegar en nuestro filtro el resultado de nuestra consulta y ¡listo!

```
-LDAPFilter
'(&(objectCategory=computer)(name=win*)(objectCategory=computer)
(userAccountControl:1.2.840.113556.1.4.803:=2))'
```

```
PS > Get-ADObject -LDAPFilter
'(&(objectCategory=computer)(name=win*)(objectCategory=computer)
(userAccountControl:1.2.840.113556.1.4.803:=2))'
```

```
DistinguishedName : CN=WINXP,CN=Computers,DC=powershell-scripting,DC=com
Name              : WINXP
ObjectClass       : computer
ObjectGUID        : 483cc19d-2bb5-4582-b053-c32095f9cfd2
```

c. Creación de un filtro genérico

Al igual que el parámetro **`-LDAPFilter`**, el parámetro **`-Filter`** recibe como entrada una cadena de caracteres. La particularidad de esta cadena que constituirá el filtro es que puede ser muy sofisticada. En efecto, su sintaxis es la del lenguaje de expresiones PowerShell. También utiliza la notación BNF (*Backus-Naur Form*). Esta notación permite describir las reglas sintácticas de los lenguajes de programación; es lo que se denomina un métalenguaje.

Como sería demasiado largo entrar en detalles, ilustraremos los filtros genéricos únicamente mediante ejemplos.

Si desea aprender más sobre la constitución de filtros genéricos, consulte en la ayuda **about_ActiveDirectory_Filter**.

Ejemplo 1:

Obtener todos los objetos del Active Directory.

```
PS > Get-ADObject -Filter *
```

Ejemplo 2:

Obtener todas la cuentas de equipos del AD.

Hemos especificado el contenido del filtro entre llaves para adecuarnos al formato BNF, pero las comillas también funcionan.

```
PS > Get-ADObject -Filter {objectClass -eq 'computer'}

name        ObjectClass ObjectGUID
----        ----------- ----------
W2K8R2VM    computer    1f9b3e06-9610-4f9e-8e62-b448f09e40f9
WIN7_US_X64 computer    0558ba9c-a181-44af-a993-1ab56770efd9
WINXP       computer    483cc19d-2bb5-4582-b053-c32095f9cfd2
WIN2K8X64   computer    58052f95-9e16-4dbc-bdcb-bce0e7ee63f1
W2K3R2SRV   computer    46652166-eeb4-4313-b178-3e9c3863653d
WINXPPSV2   computer    2a2bdd5a-6ccd-4cce-8c05-6a00cc01a8c7
```

Por razones de presentación, hemos eliminado voluntariamente la propiedad `DistinguishedName`.

Ejemplo 3:

Obtener la lista de las cuentas de equipos que empiecen por Win.

```
PS > Get-ADObject -Filter {(objectClass -eq 'computer')
-and (name -like 'Win*')}

name        ObjectClass ObjectGUID
----        ----------- ----------
WIN7_US_X64 computer    0558ba9c-a181-44af-a993-1ab56770efd9
WINXP       computer    483cc19d-2bb5-4582-b053-c32095f9cfd2
WIN2K8X64   computer    58052f95-9e16-4dbc-bdcb-bce0e7ee63f1
```

```
WINXPPSV2    computer    2a2bdd5a-6ccd-4cce-8c05-6a00cc01a8c7
```

Ejemplo 4:

Obtener la lista de las cuentas de equipos desactivados que empiecen por Win.

```
PS > $f = {(objectClass -eq 'computer') -and
           (name -like 'win*') -and (enabled -eq 'False')}
PS > Get-ADComputer -Filter $f

DistinguishedName : CN=WINXP,CN=Computers,DC=powershell-scripting,DC=com
DNSHostName       : winxp.powershell-scripting.com
Enabled           : False
Name              : WINXP
ObjectClass       : computer
ObjectGUID        : 483cc19d-2bb5-4582-b053-c32095f9cfd2
SamAccountName    : WINXP$
SID               : S-1-5-21-1005862844-2131066483-1759542542-1106
UserPrincipalName :
```

Observe que esta vez hemos utilizado el commandlet `Get-ADComputer` en lugar de `Get-ADObject`. `Get-ADComputer`, por que al ser más específico aporta propiedades específicas de los objetos de tipo Computer, como la propiedad `Enabled`.

Por consiguiente, esto nos permite simplificar nuestra consulta ya que no necesitamos especificar la clase de objeto:

```
PS > $f = {(name -like 'Win*') -and (enabled -eq 'False')}
PS > Get-ADComputer -Filter $f
```

d. Creación de un filtro basado en una identidad

Los filtros basados en una identidad permiten dirigirse directamente a un objeto individual cuya identidad ya es conocida. Esto es muy útil ya que evita tener que efectuar una amplia consulta que nos devolvería una gran cantidad de resultados y que nos veríamos obligados a filtrar. Se trata pues de una manera eficaz para establecer una conexión a un objeto conocido. O, en otras palabras, a un objeto cuya identidad conocemos.

Conviene saber que cada tipo de objeto definido en Active Directory posee atributos de identidad propios. Por ejemplo, un objeto de tipo «cuenta de equipo (ADComputer)» está caracterizado por los atributos siguientes:

Atributo de identificación	Descripción
`DistinguishedName`	Identificador LDAP, con formato: `CN=nombre,CN=Unidad_Organizativa,DC=Dominio`
`ObjectGUID`	Identificador Active Directory, con formato: `c829051e-dea9-4245-b373-d381b9181cc9`
`SID`	Identificador de seguridad, con formato: `S-1-5-21-1005862844-2131066483-1759542542-1137`
`SAMAccountName`	Identificador de cuenta, con formato: `NombreDeCuenta$`

Pero posee también otros atributos ya que un objeto cuenta de equipo hereda de la clase ADAccount, que hereda de la clase ADPrincipal, que a su vez hereda de ADObject, etc.

Veamos la jerarquía de clase definida en un modelo objeto del módulo Active Directory:

```
ADEntity
   ADRootDSE
   ADObject
     ADFineGrainedPasswordPolicy
     ADOptionalFeature
     ADOrganizationalUnit
     ADPartition
       ADDomain
     ADPrincipal
       ADAccount
         ADComputer
         ADServiceAccount
         ADUser
       ADGroup
   ADDefaultDomainPasswordPolicy
   ADForest
   ADDirectoryServer
     ADDomainController
```

Empecemos por observar los atributos de identidad de una cuenta de equipo:

```
PS > Get-ADComputer -Identity 'c829051e-dea9-4245-b373-d381b9181cc9'

DistinguishedName : CN=EXCH2K10SRV,CN=Computers,DC=powershell-scripting,
                    DC=com
DNSHostName       : EXCH2K10SRV.powershell-scripting.com
Enabled           : True
Name              : EXCH2K10SRV
ObjectClass       : computer
ObjectGUID        : c829051e-dea9-4245-b373-d381b9181cc9
SamAccountName    : EXCH2K10SRV$
SID               : S-1-5-21-1005862844-2131066483-1759542542-1137
UserPrincipalName :
```

Los atributos constituyen las propiedades del objeto devuelto. Podemos destacar que los atributos `DistinguishedName`, `ObjectGUID`, `SID` y `SamAccountName` están presentes. Los otros son por tanto atributos heredados. En todo caso, cada uno de estos atributos no heredados puede utilizarse para identificar un objeto de tipo ADComputer.

Por ejemplo, las formas siguientes nos conducen exactamente al mismo resultado que anteriormente:

```
Get-ADComputer -Identity 'EXCH2K10SRV'
Get-ADComputer -Identity 'S-1-5-21-1005862844-2131066483-1759542542-1137'
Get-ADComputer -Identity 'CN=EXCH2K10SRV,CN=Computers,DC=powershell-
scripting,DC=com'
```

2. Administración de los usuarios

Ahora que sabemos cómo efectuar búsquedas basadas en los filtros o basadas en las identidades conocidas, procederemos a entrar de lleno en el tema.

Para la administración de los usuarios, disponemos de un conjunto de comandos que podemos obtener de la manera siguiente:

```
PS > Get-Command -Module ActiveDirectory -Name *user*
```

Comando	Descripción
`Get-ADUser`	Obtiene uno o varios usuarios Active Directory.
`Set-ADUser`	Modifica un usuario Active Directory.
`New-ADUser`	Crea un usuario Active Directory.
`Remove-ADUser`	Elimina un usuario Active Directory.
`Get-ADUserResultantPasswordPolicy`	Obtiene la política de contraseña resultante para un usuario Active Directory.

La obtención de uno o varios usuarios se realiza con el comando `Get-ADUser`. Este posee los parámetros siguientes:

Parámetro	Descripción
`Filter <String>`	Especifica una cadena de consulta de tipo «filtro genérico» que recupera objetos Active Directory.
`LDAPFilter <String>`	Especifica una cadena de consulta LDAP utilizada para filtrar objetos Active Directory.
`Identity <ADUser>`	Especifica un objeto Active Directory proporcionando uno de sus valores de propiedad que permitirá identificarlo.
`ResultPageSize <Int>`	Especifica el número de objetos a incluir en una página para una consulta. Por defecto, 256 objetos por página.
`ResultSetSize <Int>`	Especifica el número máximo de objetos a devolver por una consulta. El valor por defecto es `$null`. Éste corresponderá al valor «todos los objetos».
`SearchBase <String>`	Especifica una ruta de acceso Active Directory en la que efectuar las búsquedas.
`SearchScope {Base | OneLevel | Subtree}`	Especifica el ámbito de una búsqueda Active Directory. Una consulta *Base* sólo efectúa las búsquedas en la ruta de acceso o en el objeto actual. Una consulta *OneLevel* efectúa las búsquedas en los hijos directos de la ruta de acceso o de este objeto. Una consulta *Subtree* efectúa las búsquedas en la ruta de acceso o en el objeto actual y en todos los hijos de esta ruta de acceso o de este objeto. El ámbito por defecto es *Subtree*.
`Partition <String>`	Especifica el nombre único de una partición Active Directory.
`Properties <String[]>`	Especifica las propiedades del objeto de salida a recuperar. Utilice este parámetro para recuperar las propiedades que no están incluidas en le conjunto por defecto. Especifica las propiedades de este parámetro en el formato de una lista de nombres separados por comas. Para visualizar todos los atributos definidos sobre el objeto, especifique «*».
`Server <String>`	Especifica la instancia de servicios Active Directory a la que conectarse. Esta puede ser de tipo AD DS, AD LDS o AD Snapshot. Es el dominio del equipo que ejecuta la consulta el que es escogido por defecto.
`Credential <PSCredential>`	Especifica la información de identificación de la cuenta de usuario a utilizar para efectuar esta tarea.

`AuthType {Negotiate | Basic}`	Especifica el método de autentificación a utilizar. El tipo por defecto es Negociate.

a. Obtener la lista de los usuarios

La forma más sencilla de efectuar una búsqueda de usuarios a través de todo el servicio de directorio Active Directory es la siguiente:

```
PS > Get-ADUser -Filter *
```

Si queremos únicamente listar los usuarios de una unidad organizativa, entonces escogeremos la línea de comandos siguiente:

```
PS > Get-ADUser -Filter * -SearchBase 'OU=Finanzas,DC=POWERSHELL-
SCRIPTING,DC=COM'
```

Para obtener una lista fácilmente interpretable de usuarios, será útil formatear el resultado en formato de tabla, como a continuación:

```
PS > Get-ADUser -Filter * | Format-Table GivenName,Surname, Name, Sam*

GivenName  Surname    Name              SamAccountName
---------  -------    ----              --------------
                      Administrator     Administrator
                      Guest             Guest
                      krbtgt            krbtgt
Oscar      Vázquez    Oscar Vázquez     VazquezO
```

b. Creación de usuarios

En lo que respecta a la creación de usuarios, el comando a utilizar es `New-ADUser`. Éste posee numerosos parámetros. Posee casi tantos parámetros como propiedades posee un objeto usuario (tipo ADUser).

Para conocer todos los parámetros de `New-ADUser` y su rol asociado, teclee el comando:

```
PS > Help New-ADUser -Full
```

Visto el impresionante número de parámetros que puede tener este comando, citaremos únicamente los más comunes:

Parámetro	Descripción
`SamAccountName <String>`	Especifica el nombre de la cuenta de seguridad SAM (nombre de grupo anterior a Windows 2000).
`Name <String>`	Especifica el nombre del objeto.
`Surname <String>`	Especifica el apellido del usuario.
`DisplayName <String>`	Especifica el nombre a mostrar del objeto.
`GivenName <String>`	Especifica le nombre propio del usuario.
`Description <String>`	Especifica una descripción del objeto.

`EmailAddress <String>`	Especifica el correo electrónico del usuario.
`Enabled {$true | $false}`	Especifica si la cuenta debe estar activa. Una cuenta activa requiere una contraseña.
`AccountPassword <SecureString>`	Especifica una nueva contraseña para una cuenta. Este valor se almacena en formato de cadena segura.
`CannotChangePassword {$true | $false}`	Especifica si la contraseña de la cuenta puede modificarse.
`ChangePasswordAtLogon {$true | $false}`	Especifica si una contraseña debe modificarse en el próximo inicio de sesión.
`PasswordNeverExpires {$true | $false}`	Especifica si la contraseña del usuario puede expirar.
`PasswordNotRequired {$true | $false}`	Especifica si la cuenta requiere una contraseña. Una contraseña no es necesaria en una cuenta nueva.
`HomeDirectory <String>`	Especifica el directorio de base de un usuario.
`HomeDrive <String>`	Especifica una unidad asociada a una ruta UNC definida por la propiedad `HomeDirectory`.
`ProfilePath <String>`	Especifica la ruta de acceso al perfil del usuario.
`Path <String>`	Especifica la ruta de acceso X.500 de la unidad organizativa o del contenedor donde se ha creado el nuevo objeto. Si no se informa este valor, con AD DS los objetos usuarios se ubicarán en la unidad organizativa Users.
`PassThru <Switch>`	Devuelve el nuevo objeto creado. Por defecto (es decir si -PassThru no se especifica), este comando no genera ninguna salida.
`ScriptPath <String>`	Especifica una ruta de acceso al script de inicio de sesión del usuario.
`Instance <ADUser>`	Especifica una instancia de un objeto usuario para utilizar como modelo para un nuevo objeto usuario.
`Credential <PSCredential>`	Especifica la información de identificación de la cuenta de usuario a utilizar para efectuar esta tarea.

Para crear una cuenta de usuario, es preciso como mínimo precisar la propiedad `Name`.

Ejemplo:

Creación minimalista de un usuario.

```
PS > New-ADUser -Name JoseBar
```

Esta línea de comandos crea un usuario en el contenedor «Users» (ya que no hemos precisado nada) que aparecerá con el nombre de «JoseBar». Tendrá igualmente por defecto un atributo `SamAccountName` con el valor «JoseBar».

Para verificar la creación y obtener nuestro usuario, podemos crear un filtro sobre su nombre:

```
PS > Get-ADUser -Filter {(name -like 'jose*')} | Format-Table Name, Sam*, SID

Name     SamAccountName SID
```

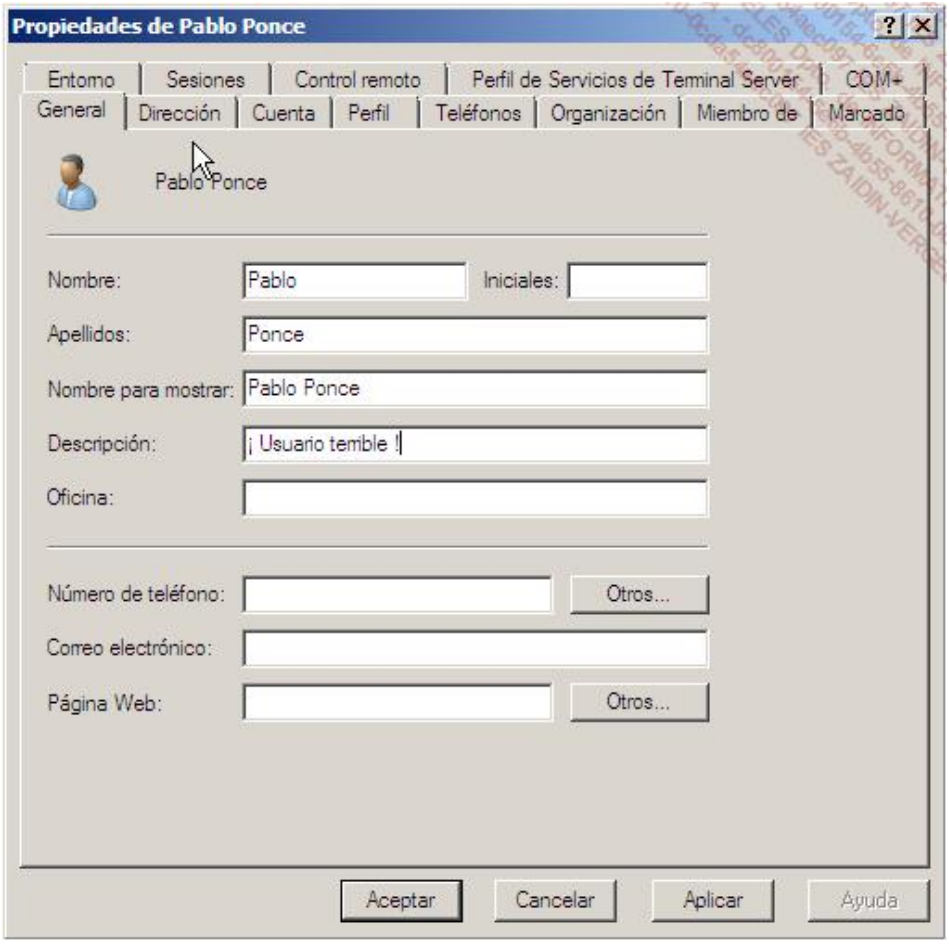
```
----     -------------- ---
JoseBar  JoseBar        S-1-5-21-1005862844-2131066483-1759542542-1138
```

Acabamos de efectuar una búsqueda en todas las cuentas cuyo nombre comience por «jose» y hemos aprovechado de paso para mostrar su SID.

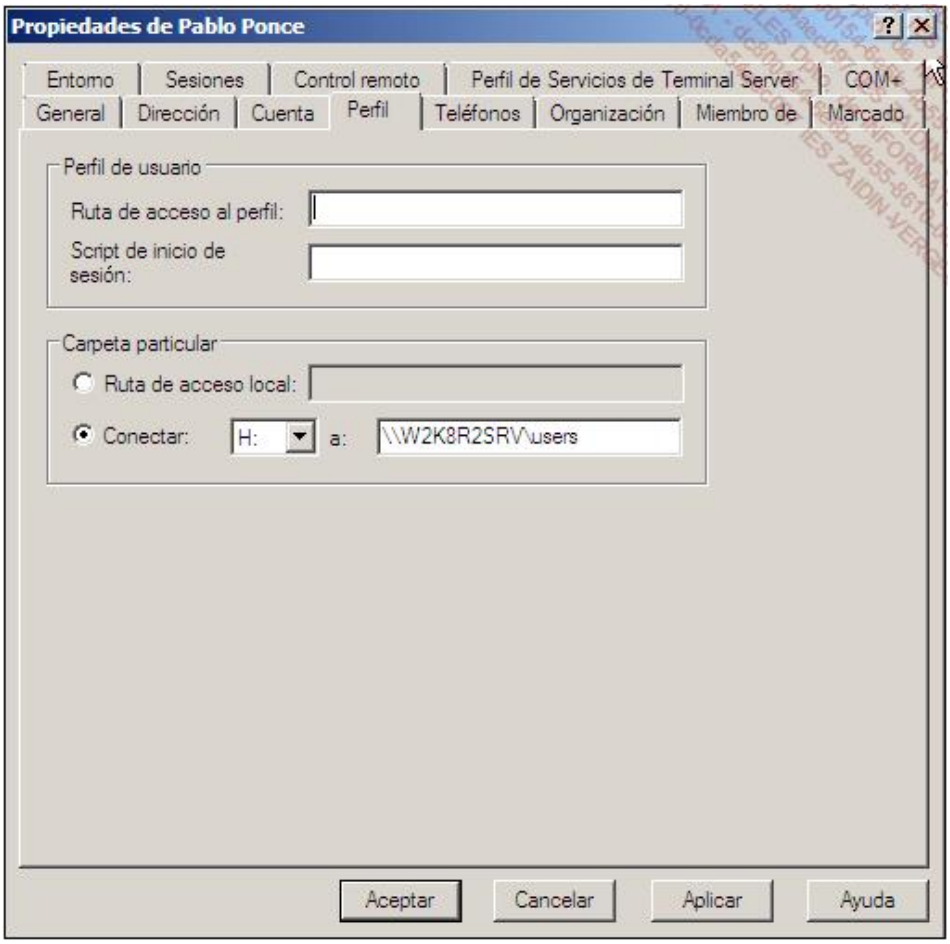
Si ahora queremos crear una cuenta con más atributos y queremos colocarla en la unidad organizativa «Finanzas», podemos realizar lo siguiente:

```
PS > New-ADUser -SamAccountName Ponce -Name 'Pablo Ponce' `
           -GivenName Pablo -Surname Ponce                    `
           -DisplayName 'Pablo Ponce' -description '¡Usuario terrible!' `
           -HomeDrive 'H:' -HomeDirectory '\\W2K8R2SRV\users'             `
           -Path 'OU=Finanzas,dc=Powershell-scripting,dc=com'
```

Este es el resultado obtenido cuando mostramos las propiedades de nuestro nuevo usuario, a través de la consola «Usuarios y equipos de Active Directory»:



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



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

c. Asignar una contraseña al crearla

Podemos asignar una contraseña en el momento de la creación de una cuenta, con el comando `New-ADUser`. Si por el contrario, queremos modificar la contraseña de una cuenta existente, entonces debemos utilizar el comando `Set-ADAccountPassword`.

Sea como sea, puesto que el valor esperado para el parámetro **`-AccountPassword`** es de tipo cadena segura (SecureString), debemos efectuar algunas pequeñas manipulaciones suplementarias:

```
PS > $passwd = 'Passw0rd123*!'
PS > $passwd = ConvertTo-SecureString $passwd -AsPlainText -Force
PS > New-ADUser -SamAccountName DuranteJC -Name 'JC Durante' `
-AccountPassword $passwd
```

d. Asignar una contraseña a una cuenta existente

Si la cuenta ya existe, entonces en este caso tendremos que hacer lo siguiente:

```
PS > Set-ADAccountPassword -Identity DuranteJC -NewPassword $passwd -Reset
```

Aunque lógicamente hubiéramos podido imaginar que el comando `Set-ADUser` podría hacerlo, resulta que no permite el cambio de contraseña. Para ello, será necesario utilizar el comando `Set-ADAccountPassword`. Este permite definir una contraseña para un usuario, un equipo o una cuenta de servicio.

`Set-ADAccountPassword` posee los parámetros siguientes:

Parámetro	Descripción
`Identity <ADAccount>`	Especifica un objeto Active Directory proporcionando uno de sus valores de propiedad que permitirá identificarlo.
`AuthType {Negotiate | Basic}`	Especifica el método de autentificación a utilizar. El tipo por defecto es `Negociate`.
`Credential <PSCredential>`	Especifica la información de identificación de la cuenta de usuario a utilizar para efectuar esta tarea.
`NewPassword <SecureString>`	Especifica un valor de la nueva contraseña.
`OldPassword <SecureString>`	Especifica el valor de la contraseña más reciente.
`Partition <String>`	Especifica el nombre único de una partición Active Directory.
`PassThru <Switch>`	Devuelve el nuevo objeto creado. Por defecto (es decir si `-PassThru` no se especifica), este comando no genera ninguna salida.
`Reset <Switch>`	Especifica la reinicialización de la contraseña en una cuenta. Cuando utilice este parámetro, deberá definir el parámetro NewPassword. No será obligatorio especificar el parámetro OldPassword.
`Server <String>`	Especifica la instancia de servicios Active Directory a la que conectarse. Esta puede ser de tipo AD DS, AD LDS o AD Snapshot. El dominio del equipo que ejecuta la consulta es el que es escogido por defecto.

e. Activar una cuenta al crearla

Para activar una cuenta en el momento de crearla, es necesario asignarle una contraseña. Esta contraseña deberá estar en consonancia con la política de seguridad de su dominio.

Para crear una cuenta activa, siga el ejemplo siguiente:

```
PS > $passwd = 'Passw0rd123*!'
PS > $passwd = ConvertTo-SecureString $passwd -AsPlainText -Force

PS > New-ADUser -SamAccountName Ponce -Name 'Pablo Ponce' `
           -GivenName Pablo -Surname Ponce                    `
           -DisplayName 'Pablo Ponce' -description '¡ Usuario terrible !' `
           -HomeDrive 'H:' -HomeDirectory '\\W2K8R2SRV\users' `
           -Path 'OU=Finanzas,DC=Powershell-scripting,dc=com'  `
           -AccountPassword $passwd -Enabled $true
```

f. Activar una cuenta existente

Para activar una cuenta existente, es preciso proceder en dos etapas. La primera consistirá en asignar una contraseña, y la segunda en activar la cuenta.

```
PS > Set-ADAccountPassword -Identity LópezE -NewPassword $passwd
PS > Set-ADUser -Identity LópezE -Enabled $true
```

g. Leer uno o varios atributos

Para leer un atributo o propiedad de un usuario, será necesario en primer lugar conectarse al objeto del servicio de directorio correspondiente. Para ello, utilizaremos el comando `Get-ADUser` con el parámetro **`-Identity`**. Acto seguido, no necesitaremos más que solicitar la o las propiedades de nuestra elección pasándolas al parámetro **`-Properties`**.

Ejemplo:

```
PS > Get-ADUser -Identity Ponce `
-Properties Name,WhenCreated,PasswordLastSet

DistinguishedName : CN=Pablo Ponce,OU=Finanzas,DC=powershell-
                    scripting,DC=com
Enabled           : True
GivenName         : Pablo
Name              : Pablo Ponce
ObjectClass       : user
ObjectGUID        : 27fe799d-d759-464e-b253-20847a6e7b6a
PasswordLastSet   : 31/10/2009 17:22:14
SamAccountName    : Ponce
SID               : S-1-5-21-1005862844-2131066483-1759542542-1141
Surname           : Ponce
UserPrincipalName :
WhenCreated       : 31/10/2009 17:22:14
```

Por defecto, nos encontramos con todos los otros atributos. Para filtrar, podemos formatear nuestro resultado de forma que obtengamos únicamente las propiedades que nos interesan.

```
PS > Get-ADUser -Identity Ponce -Properties * |
FT Name,WhenCreated,PasswordLastSet
```

Para abreviar la línea de comandos, observe que hemos utilizado el alias «FT» en lugar del comando `Format-Table`.

Si esta vez sólo queremos recuperar una única propiedad, por ejemplo `WhenCreated`, y almacenarla en una variable podemos escribir lo siguiente:

```
PS > $dateCreation =
(Get-ADUser -Identity Ponce -Properties WhenCreated).WhenCreated
```

Observemos el contenido de nuestra variable `$dateCreation`:

```
PS > $dateCreation
sabado 31 octubre 2009 17:22:14
```

Podemos observar que la fecha obtenida presenta un formato diferente al anterior; esto demuestra que PowerShell manipula los objetos y no el texto. Si el formato de la fecha no le conviene, no dude en consultar el apartado que trata sobre la manipulación de las fechas en el capítulo Control del Shell para darle el formato deseado.

h. Obtener todos los atributos

Obtener todas las propiedades que un objeto Usuario posee, es tan sencillo como esto:

```
PS > Get-ADUser -Identity Ponce -Properties *
```

```
AccountExpirationDate             :
accountExpires                    : 9223372036854775807
AccountLockoutTime                :
AccountNotDelegated               : False
AllowReversiblePasswordEncryption : False
BadLogonCount                     : 0
badPasswordTime                   : 0
badPwdCount                       : 0
CannotChangePassword              : False
CanonicalName                     : powershell-scripting.com/Finanzas/
                                    Pablo Ponce
Certificates                      : {}
City                              :
CN                                : Pablo Ponce
codePage                          : 0
Company                           :
Country                           :
countryCode                       : 0
Created                           : 31/10/2009 17:22:14
createTimeStamp                   : 31/10/2009 17:22:14
Deleted                           :
Department                        :
Description                       : ¡Usuario terrible!
DisplayName                       : Pablo Ponce
DistinguishedName                 : CN=Pablo Ponce,OU=Finanzas,
                                    DC=powershell-scripting,DC=com
Division                          :
DoesNotRequirePreAuth             : False
dSCorePropagationData             : {01/01/1601 01:00:00}
EmailAddress                      :
EmployeeID                        :
EmployeeNumber                    :
Enabled                           : True
Fax                               :
GivenName                         : Pablo
HomeDirectory                     : \\W2K8R2SRV\users
HomedirRequired                   : False
HomeDrive                         : H:
HomePage                          :
HomePhone                         :
Initials                          :
instanceType                      : 4
isDeleted                         :
LastBadPasswordAttempt            :
LastKnownParent                   :
lastLogoff                        : 0
lastLogon                         : 0
LastLogonDate                     :
LockedOut                         : False
logonCount                        : 0
LogonWorkstations                 :
Manager                           :
MemberOf                          : {}
MNSLogonAccount                   : False
MobilePhone                       :
```

```
Modified                          : 31/10/2009 17:22:14
modifyTimeStamp                   : 31/10/2009 17:22:14
msDS-User-Account-Control-Computed : 0
Name                              : Pablo Ponce
nTSecurityDescriptor              : System.DirectoryServices.
                                    ActiveDirectorySecurity
ObjectCategory                    : CN=Person,CN=Schema,CN=Configuration,
                                    DC=powershell-scripting,DC=com
ObjectClass                       : user
ObjectGUID                        : 27fe799d-d759-464e-b253-20847a6e7b6a
objectSid                         : S-1-5-21-1005862844-2131066483-
                                    1759542542-1141
Office                            :
OfficePhone                       :
Organization                      :
OtherName                         :
PasswordExpired                   : False
PasswordLastSet                   : 31/10/2009 17:22:14
PasswordNeverExpires              : False
PasswordNotRequired               : False
POBox                             :
PostalCode                        :
PrimaryGroup                      : CN=Domain Users,CN=Users,
                                    DC=powershell-scripting,DC=com
primaryGroupID                    : 513
ProfilePath                       :
ProtectedFromAccidentalDeletion   : False
pwdLastSet                        : 129014797346100371
SamAccountName                    : Ponce
sAMAccountType                    : 805306368
ScriptPath                        :
sDRightsEffective                 : 15
ServicePrincipalNames             : {}
SID                               : S-1-5-21-1005862844-2131066483-
                                    1759542542-1141
SIDHistory                        : {}
SmartcardLogonRequired            : False
sn                                : Ponce
State                             :
StreetAddress                     :
Surname                           : Ponce
Title                             :
TrustedForDelegation              : False
TrustedToAuthForDelegation        : False
UseDESKeyOnly                     : False
userAccountControl                : 512
userCertificate                   : {}
UserPrincipalName                 :
uSNChanged                        : 17991
uSNCreated                        : 17987
whenChanged                       : 31/10/2009 17:22:14
whenCreated                       : 31/10/2009 17:22:14
```

i. Modificar un atributo

La modificación de un atributo se realiza con el comando `Set-ADUser`. Por ejemplo, si queremos modificar el atributo `SamAccountName`, utilizaremos el parámetro del mismo nombre. Como mostramos a continuación:

```
PS > Set-ADUser -Identity Ponce -SamAccountName García
```

Podemos también utilizar el parámetro **`-Replace`** para modificar una o varias propiedades en una única operación.

Ejemplo:

Modificación de la descripción, del número de teléfono principal y de otros números de teléfono:

```
PS > Set-ADUser -Identity Ponce -Replace @{
     Description = 'Usuario simpático' ;
     TelephoneNumber = '0110203040';
     OtherTelephone = @('0250403020','0340506070')}
```

Observe que hemos pasado un valor de tipo tabla a la propiedad `OtherTelephone`; así podemos asignarle diferentes valores de una sola vez.

j. Borrar un atributo

Esta vez es la propiedad `Clear` la que entrará en escena. Eliminaremos los números de teléfono secundarios del usuario «Ponce», como se muestra a continuación:

```
PS > Set-ADUser -Identity Ponce -Clear OtherTelephone
```

k. Eliminar un usuario

Hemos visto que en el conjunto de comandos utilizados para la administración de los usuarios existía el comando `Remove-ADUser`.

Su utilización es bien sencilla, únicamente necesita pasarle a su parámetro `Identity` un objeto de tipo ADUser; como en el ejemplo siguiente:

```
PS > Remove-ADUser -Identity Ponce

Confirmar
¿Está usted seguro de querer efectuar esta acción?
Operación «Remove» en curso sobre «CN=Pablo Ponce,OU=Finanzas,
DC=powershell-scripting,DC=com».
[S] Sí  [T] Sí a todo  [N] No  [O] No a todo  [U] Suspender  [?]
Ayuda (el valor por defecto es «S»):
```

El funcionamiento por defecto de este comando pide una confirmación. Si desea evitar esta confirmación, deberá hacer lo siguiente:

```
PS > Remove-ADUser -Identity Ponce -Confirm:$false
```

3. Administración de los grupos

Para la administración de los grupos y de su contenido, disponemos de una decena de comandos. Podemos obtenerlos del modo siguiente:

```
PS > Get-Command -Module ActiveDirectory -Name *group*
```

Comando	Descripción
`Add-ADGroupMember`	Añade uno o varios miembros a un grupo Active Directory.
`Add-ADPrincipalGroupMembership`	Añade un miembro a uno o varios grupos Active Directory.
`Get-ADAccountAuthorizationGroup`	Obtiene los grupos de seguridad para el usuario, el equipo o la cuenta de servicio especifica.
`Get-ADGroup`	Obtiene uno o varios grupos Active Directory.
`Get-ADGroupMember`	Obtiene los miembros de un grupo Active Directory.
`Get-ADPrincipalGroupMembership`	Obtiene los grupos Active Directory que posee un usuario, un equipo, un grupo o una cuenta de servicio especificada.
`New-ADGroup`	Crea un grupo Active Directory.
`Remove-ADGroup`	Elimina un grupo Active Directory.
`Remove-ADGroupMember`	Elimina uno o varios miembros de un grupo Active Directory.
`Remove-ADPrincipalGroupMembership`	Elimina un miembro de uno o varios grupos Active Directory.
`Set-ADGroup`	Modifica un grupo Active Directory.

a. Listar los grupos

Para obtener la lista de los grupos presentes en Active Directory, utilizaremos el commandlet `Get-ADGroup`. Éste posee los mismos parámetros que el comando `Get-ADUser` que son los siguientes:

Parámetro	Descripción
`Filter <String>`	Especifica una cadena de consulta de tipo «filtro genérico» que recupera objetos Active Directory.
`LDAPFilter <String>`	Especifica una cadena de consulta LDAP utilizada para filtrar objetos Active Directory.
`Identity <ADUser>`	Especifica un objeto Active Directory proporcionando uno de sus valores de propiedad que permitirá identificarlo.
`ResultPageSize <Int>`	Especifica el número de objetos a incluir en una página para una consulta. Por defecto, 256 objetos por página.
`ResultSetSize <Int>`	Especifica el número máximo de objetos a devolver por una consulta. El valor por defecto es $null. Este corresponderá al valor «todos los objetos».
`SearchBase <String>`	Especifica una ruta de acceso Active Directory en la que efectuar las búsquedas.
`SearchScope {Base | OneLevel | Subtree}`	Especifica el ámbito de una búsqueda Active Directory. Una consulta *Base* sólo efectúa las búsquedas en la ruta de acceso o en el objeto actual. Una consulta *OneLevel* efectúa las búsquedas en los hijos directos de la ruta de acceso o de este objeto. Una consulta *Subtree* efectúa las búsquedas en la ruta de acceso o en el

	objeto actual y en todos los hijos de esta ruta de acceso o de este objeto. El ámbito por defecto es *Subtree*.
`Partition <String>`	Especifica el nombre único de una partición Active Directory.
`Properties <String[]>`	Especifica las propiedades del objeto de salida a recuperar. Utiliza este parámetro para recuperar las propiedades que no están incluidas en el conjunto por defecto. Especifica las propiedades de este parámetro en el formato de una lista de nombres separados por comas. Para visualizar todos los atributos definidos sobre el objeto, especifique «*».
`Server <String>`	Especifica la instancia de servicios Active Directory a la que conectarse. Esta puede ser de tipo AD DS, AD LDS o AD Snapshot. El dominio del equipo que ejecuta la consulta es el que es escogido por defecto.
`Credential <PSCredential>]`	Especifica la información de identificación de la cuenta de usuario a utilizar para efectuar esta tarea.
`AuthType {Negotiate | Basic}`	Especifica el método de autentificación a utilizar. El tipo por defecto es Negociate.

Obtener todos los grupos del dominio

La liste íntegra de los grupos puede obtenerse de la forma siguiente:

```
PS > Get-ADGroup -Filter *
```

Para hacer que esta lista sea un poco más agradable a la vista es interesante mostrarla en forma de tabla; en este caso es preciso indicar algunos nombres de propiedades:

```
PS > Get-ADGroup -Filter * | Format-Table Name, GroupScope

Name                                GroupScope
----                                ----------
Administrators                     DomainLocal
Users                              DomainLocal
Guests                             DomainLocal
Print Operators                    DomainLocal
Backup Operators                   DomainLocal
Remote Desktop Users               DomainLocal
IIS_IUSRS                          DomainLocal
...
Event Log Readers                  DomainLocal
Domain Computers                        Global
Domain Controllers                      Global
Schema Admins                     Universal
Enterprise Admins                 Universal
Domain Admins                           Global
Domain Users                            Global
Domain Guests                           Global
Group Policy Creator Owners             Global
...
```

Obtener los grupos de un cierto tipo

De la misma forma, para obtener únicamente los grupos universales, la creación de un filtro mejorado cumplirá

perfectamente su función:

```
PS > Get-ADGroup -Filter {GroupScope -eq 'Universal'} |
Format-Table Name,GroupScope

Name                                Groupscope
----                                ----------
Schema Admins                        Universal
Enterprise Admins                    Universal
Enterprise Read-only Domain Controllers  Universal
```

Obtener los grupo de una unidad organizativa determinada

Para no recuperar todos los grupos del dominio, especificando un parámetro `SearchBase` podremos restringir el ámbito de la búsqueda a una unidad organizativa determinada, como mostramos a continuación:

```
PS > Get-ADGroup -Filter * -SearchBase 'OU=finanzas,DC=powershell-scripting,
DC=com'
```

Obtener un grupo determinado

Y por supuesto, si queremos recuperar únicamente un grupo determinado, será práctico utilizar el parámetro `Identity`, como se muestra a continuación:

```
PS > Get-ADGroup -Identity Administrators

DistinguishedName : CN=Administrators,CN=Builtin,DC=powershell-scripting,
                    DC=com
GroupCategory     : Security
GroupScope        : DomainLocal
Name              : Administrators
ObjectClass       : group
ObjectGUID        : 9cabb43d-ecfa-4cb0-910d-33c9d9490127
SamAccountName    : Administrators
SID               : S-1-5-32-544
```

b. Creación de grupos

La creación de los grupos se efectúa con la ayuda del comando `New-ADGroup`. Éste comprende los parámetros siguientes:

Parámetro	Descripción
`Name <String>`	Especifica el nombre del objeto.
`Description <String>`	Especifica una descripción del objeto.
`DisplayName <String>`	Especifica el nombre a mostrar del objeto.
`GroupCategory {Distribution | Security}`	Especifica la categoría del grupo.
`GroupScope {DomainLocal | Global | Universal}`	Especifica el ámbito del grupo.

SamAccountName <String>	Especifica el nombre de la cuenta de seguridad SAM (nombre del grupo anterior a Windows 2000).
HomePage <String>	Especifica la URL de la página de inicio del objeto.
Instance <ADGroup>	Especifica una instancia de un objeto grupo para utilizar como modelo para un nuevo objeto grupo.
ManagedBy <ADPrincipal>	Especifica el usuario o el grupo que administra el objeto.
OtherAttributes <hashtable>	Especifica los valores de los atributos de objetos para los atributos que no están representados por los parámetros.
PassThru <Switch>	Devuelve el nuevo objeto creado. Por defecto (es decir si -PassThru no se especifica), este comando no genera ninguna salida.
Path <String>	Especifica la ruta de acceso X.500 de la unidad organizativa o del contenedor donde se ha creado el nuevo objeto.
Server <String>	Especifica la instancia de servicios Active Directory a la que conectarse. Ésta puede ser de tipo AD DS, AD LDS o AD Snapshot. Es el dominio del equipo que ejecuta la consulta el que es escogido por defecto.
AuthType {Negotiate | Basic}	Especifica el método de autentificación a utilizar. El tipo por defecto es Negociate.
Credential <PSCredential>	Especifica la información de identificación de la cuenta de usuario a utilizar para efectuar esta tarea.

Para crear un grupo, será necesario por lo mínimo asignarle un nombre (propiedad `Name`) y un ámbito (`GroupScope`).

Si no se especifica ninguna categoría (`GroupCategory`) entonces por defecto será creado como un grupo de seguridad. Si no se especifica el `SamAccountName` entonces esta propiedad tomará el mismo nombre que la propiedad Name del grupo.

Ejemplo:

Creación de un grupo local de seguridad

```
PS > New-ADGroup -Name UsuariosVDI -GroupScope DomainLocal `
-GroupCategory Security
```

Creación de un grupo global de distribución

```
PS > New-ADGroup -Name UsuariosTS -GroupScope Global `
-GroupCategory Distribution
```

Creación de un grupo universal de seguridad en la unidad organizativa «Finanzas» y delegación de la administración del grupo a un usuario:

```
PS > New-ADGroup -Name GrpDelega1 -GroupScope universal `
-GroupCategory security `
  -ManagedBy LópezE -Path 'OU=finanzas,DC=powershell-scripting,DC=com'
```

c. Enumerar los miembros de un grupo

Para obtener el contenido de un grupo emplearemos el comando `Get-ADGroupMember`. Veamos los parámetros que posee:

Parámetro	Descripción
`Identity <ADGroup>`	Especifica un objeto Active Directory proporcionando uno de sus valores de propiedad que permitirá identificarlo.
`Recursive <Switch>`	Obtiene todos los miembros del grupo especificado así como los miembros de cualquier grupo hijo.
`Partition <String>`	Especifica el nombre único de una partición Active Directory.
`Server <String>`	Especifica la instancia de servicios Active Directory a la que conectarse. Ésta puede ser de tipo AD DS, AD LDS o AD Snapshot. El dominio del equipo que ejecuta la consulta es el que es escogido por defecto.
`Credential <PSCredential>`	Especifica la información de identificación de la cuenta de usuario a utilizar para efectuar esta tarea.
`AuthType {Negotiate | Basic}`	Especifica el método de autentificación a utilizar. El tipo por defecto es Negociate.

Ejemplo: recuperación del contenido del grupo «Administrators»

```
PS > Get-ADGroupMember -Identity Administrators |
Format-Table Name,ObjectClass

Name              ObjectClass
----              -----------
Domain Admins     group
Enterprise Admins group
Administrator     user
```

Podemos constatar que recuperamos tanto objetos grupo como objetos usuario. Si ahora utilizamos el parámetro **`-Recursive`**, el resultado obtenido será el desglose del contenido de cada grupo devuelvo.

Ejemplo:

```
PS > Get-ADGroupMember -Identity Administrators -Recursive |
Format-Table Name,ObjectClass

Name          ObjectClass
----          -----------
Administrator user
AdminLópez    user
```

En este ejemplo obtenemos el usuario «AdminLópez» ya que él es miembro del grupo «Domain Admins».

Si los objetos de tipo «cuenta de equipo» hubiesen estado presentes en alguno de los grupos, «Domain Admins» o «Entreprise Admins» también se hubiesen recuperado. El parámetro Recursive tiene la función de extraer el contenido de todos los grupos.

d. Añadir miembros a un grupo (1 a 1 ó n a 1)

La adición de miembros a un grupo se efectúa con el commandlet `Add-ADGroupMember`. Los parámetros disponibles son los siguientes:

Parámetro	Descripción
`Identity <ADGroup>`	Especifica un objeto Active Directory proporcionando uno de sus valores de propiedad que permitirá identificarlo.
`Members <ADPrincipal[]>`	Especifica un conjunto de objetos (usuario, grupo y equipo), en una lista separada por comas, a añadir a un grupo.
`Partition <String>`	Especifica el nombre único de una partición Active Directory.
`Server <String>`	Especifica la instancia de servicios Active Directory a la que conectarse. Ésta puede ser de tipo AD DS, AD LDS o AD Snapshot. El dominio del equipo que ejecuta la consulta es el que es escogido por defecto.
`PassThru <Switch>`	Devuelve el nuevo objeto creado. Por defecto (es decir si -PassThru no se especifica), este comando no genera ninguna salida.
`Credential <PSCredential>`	Especifica la información de identificación de la cuenta de usuario a utilizar para efectuar esta tarea.
`AuthType {Negotiate | Basic}`	Especifica el método de autentificación a utilizar. El tipo por defecto es Negociate.

Ejemplo: añadir varios usuarios a un grupo.

```
PS > Add-ADGroupMember -Identity UsuariosVDI `
-Members LópezE,DuranteJC
```

Verifiquemos si la adición de miembros al grupo «UsuariosVDI» ha funcionado correctamente:

```
PS > Get-ADGroupMember UsuariosVDI

distinguishedName : CN=Eduardo López,OU=Finanzas,DC=powershell-scripting,
                    DC=com
name              : Eduardo López
objectClass       : user
objectGUID        : eadb78c7-0aa7-4c6b-8496-388f46f3cfea
SamAccountName    : LópezE
SID               : S-1-5-21-1005862844-2131066483-1759542542-1142

distinguishedName : CN=Juan Carlos Durante,CN=Users,DC=powershell-scripting,
                    DC=com
name              : Juan Carlos Durante
objectClass       : user
objectGUID        : 1a33625b-14bc-4ec2-b4d9-e53116980350
SamAccountName    : DuranteJC
SID               : S-1-5-21-1005862844-2131066483-1759542542-1139
```

Ningún problema, ¡funciona! Observe que hemos omitido detallar el parámetro `Identity`. Esto no constituye un problema para PowerShell ya que este parámetro es el primero esperado por el comando.

e. Añadir un miembro a uno o varios grupos (1 a 1 ó 1 a n)

El enfoque de esta operación es un poco diferente de la anterior. Para hacerse una idea, es como si usted abriese las propiedades de un usuario con la interfaz gráfica **Usuarios y equipos de Active Directory** y fuese a la pestaña **Miembro de** para añadir los grupos.

De esta forma puede añadir un usuario a muchos grupos en una sóla operación. Para hacerlo por línea de comandos, es necesario utilizar el comando `Add-ADPrincipalGroupMembership`. Éste acepta los parámetros siguientes:

Parámetro	Descripción
`Identity <ADPrincipal>`	Especifica un objeto Active Directory proporcionando uno de sus valores de propiedad que permitirá identificarlo.
`MemberOf <ADGroup[]]>`	Especifica los grupos Active Directory a los que añadir un usuario, equipo o grupo como miembro.
`Partition <String>`	Especifica el nombre único de una partición Active Directory.
`Server <String>`	Especifica la instancia de servicios Active Directory a la que conectarse. Ésta puede ser de tipo AD DS, AD LDS o AD Snapshot. El dominio del equipo que ejecuta la consulta es el que es escogido por defecto.
`PassThru <Switch>`	Devuelve el nuevo objeto creado. Por defecto (es decir si -PassThru no se especifica), este comando no genera ninguna salida.
`Credential <PSCredential>`	Especifica la información de identificación de la cuenta de usuario a utilizar para efectuar esta tarea.
`AuthType {Negotiate | Basic}`	Especifica el método de autentificación a utilizar. El tipo por defecto es Negociate.

Este comando tiene un nombre genérico ya que permite añadir a un grupo, no sólo una cuenta de usuario sino también una cuenta de equipo o un grupo. Las entidades de seguridad Active Directory tienen el nombre «principal» en la jerga del protocolo Kerberos.

Ejemplo: asignación de un usuario a un grupo

```
PS > Add-ADPrincipalGroupMembership AdminLópez `
     -MemberOf 'Enterprise Admins'
```

Ejemplo: asignación de un usuario a varios grupos

```
PS > Add-ADPrincipalGroupMembership -Identity DuranteJC `
     -MemberOf Grupo1, Grupo2, Grupo3
```

f. Eliminar uno o varios miembros de un grupo

Después de utilizar el comando `Add-ADGroupMember` para añadir uno o varios miembros a un grupo, esta vez vamos a utilizar su opuesto, el comando `Remove-ADGroupMember`.

Éste posee exactamente los mismos parámetros que su hermano `Add-ADGroupMember`. Se puede utilizar de la forma siguiente:

```
PS > Remove-ADGroupMember -Identity UsuariosVDI `
    -Members DuranteJC, LópezE

Confirmar
¿Está usted seguro de querer efectuar esta acción?
Operación «Set» en curso sobre «CN=UsuariosVDI,CN=Users,
DC=powershell-scripting,DC=com».
[S] Sí  [T] Sí a todo  [N] No  [O] No a todo  [U] Suspender  [?]
Ayuda (el valor por defecto es «S»:
```

Como puede observar, este comando pide una confirmación para ejecutarse correctamente. Si desea evitar la confirmación podrá utilizar el parámetro **`-Confirm`** de la siguiente forma:

```
PS > Remove-ADGroupMember -Identity UsuariosVDI `
    -Members DuranteJC, LópezE -Confirm:$false
```

g. Eliminar un miembro de uno o de varios grupos

La eliminación de un miembro de uno o varios grupos en una sóla operación se efectúa exactamente de la misma manera que para añadirlo a uno o varios grupos (lo que hemos hecho con el comando `Add-ADPrincipalGroupMembership`). Para ello, utilizaremos el comando `Remove-ADPrincipalGroupMembership`. Este comando presenta exactamente los mismos parámetros que su hermano gemelo.

Ejemplo: eliminación de un usuario de un grupo

```
PS > Remove-ADPrincipalGroupMembership -Identity AdminLópez `
   -MemberOf 'Enterprise Admins'

Eliminar los miembros del grupo
¿Desea eliminar todos los miembros especificados de los grupos
especificados? [S] Sí  [T] Sí a todo  [N] No  [O] No a todo  [U] Suspender
[?] Ayuda (el valor por defecto es «S»):
```

Como cada vez que se elimina un objeto del Active Directory, se pide una confirmación. Para evitarlo, podemos pasar el valor «false» al parámetro **`-Confirm`**, como se muestra aquí:

```
PS > Remove-ADPrincipalGroupMembership -Identity AdminLópez `
   -MemberOf 'Enterprise Admins' -Confirm:$false
```

Ejemplo: eliminación de un usuario de varios grupos

```
PS > Remove-ADPrincipalGroupMembership -Identity DuranteJC `
     -MemberOf Grupo1, Grupo2, Grupo3 -Confirm:$false
```

h. Eliminar un grupo

La eliminación de un grupo se realiza con el comando `Remove-ADGroup`.

Su utilización es muy sencilla, únicamente requiere que se pase al parámetro **`-Identity`** un objeto de tipo ADGroup; como en el ejemplo siguiente:

```
PS > Remove-ADGroup -Identity GrpDelega1

Confirmar
¿Está usted seguro de querer efectuar esta acción?
Operación «Remove» en curso «CN=GrpDelega1,OU=Finanzas,
DC=powershell-scripting,DC=com».
[S] Sí  [T] Sí a todo  [N] No  [O] No a todo  [U] Suspender  [?]
Ayuda (el valor por defecto es «S»):
```

Debido al funcionamiento por defecto de este commandlet, como todos los de la familia `Remove-AD*`, se nos pedirá una confirmación. Si desea evitar la confirmación, deberá hacer lo siguiente:

```
PS > Remove-ADGroup -Identity GrpDelega1 -Confirm:$false
```