Active Directory: Characters to Escape

Almost any characters can be used in distinguished names in Active Directory. However, some characters must be escaped with the backslash "\" escape character.

Table of Contents

Active Directory

Active Directory requires that the following ten characters be escaped with the backslash "\" escape character if they appear in any of the individual components of a distinguished name:

Comma	,
Backslash character	\
Pound sign (hash sign)	#
Plus sign	+
Less than symbol	<
Greater than symbol	>
Semicolon	;
Double quote (quotation mark)	"
Equal sign	=
Leading or trailing spaces	

The space character must be escaped only if it is the leading or trailing character in any component of a distinguished name. The commas that separate components in a distinguished name are not escaped. The following table shows example relative distinguished names as they would appear in Active Directory Users and Computers and the corresponding escaping required in the distinguished name:

Relative Distinguished Name	**Distinguished Name**
Smith, James K.	cn=Smith\, James K.,ou=West,dc=MyDomain,dc=com
Sales\Engineering	ou=Sales\\Engineering,dc=MyDomain,dc=com
East#Test + Lab	cn=East\#Test \+ Lab,ou=West,dc=MyDomain,dc=com
Jim Smith	cn=\ Jim Smith \ ,ou=West,dc=MyDomain,dc=com

These characters should only be escaped in distinguished names. They should not be escaped in any other Active Directory attributes, like description, givenName, or even cn.

The canonicalName attribute is similar to the distinguishedName attribute, but the escaping is different. The canonicalName is an operational attribute, also called a constructed attribute. You cannot modify this attribute.

However, the value is displayed on the "Object" tab of Active Directory Users and Computers. In the canonicalName both the forward slash and backslash characters are escaped using the backslash escape character. However, no other characters are escaped.

Some characters that are allowed in distinguished names and do not need to be escaped include:

```
* ( ) . & - _ [ ] ` ~ | @ $ % ^ ? : { } ! '
```

↑ Return to Top

ADSI

ADSI is the acronym for Active Directory Service Interfaces. This provides a library of interfaces for managing objects in Active Directory. All VBScript programs and many PowerShell scripts use ADSI interfaces. ADSI requires that the forward slash character `"/"` also be escaped in distinguished names. The ten characters in the table at the top of this page, plus the forward slash, must be escaped in most scripts where distinguished names are used to bind to objects in Active Directory.

↑ Return to Top

Non-Keyboard Characters

You can escape any characters in distinguished names, including foreign and other non-keyboard characters. Use the backslash escape character `"\"` and the two character ASCII hexadecimal representation of the character. For example:

Character in Distinguished Name	Hex Representation
á	\E1
é	\E9
í	\ED
ó	\F3
ú	\FA
ñ	\F1

↑ Return to Top

VBScript

All VBScript programs use ADSI to communicate with Active Directory. The ten characters in the table at the top of this page, plus the forward slash, must be escaped in VBScript programs where distinguished names are used to bind to objects in Active Directory.

If you use the NameTranslate object in VBScript to convert names into distinguished names, the ten characters in the table at the top of this page will be properly escaped, but not the forward slash character. If the distinguished name you retrieve with NameTranslate has the `"/"` character, you must replace it with `"\/"` before you use the value to bind to the Active Directory object to avoid an error. For example, this VBScript program converts a user sAMAccountName into the corresponding distinguished name, then binds to the user object:

```
' Constants for the NameTranslate object.
Const ADS_NAME_INITTYPE_GC = 3
```

```
Const ADS_NAME_TYPE_NT4 = 3
Const ADS_NAME_TYPE_1779 = 1


' Specify the NetBIOS name of the domain and the sAMAccountName of the user.
strNTName = "MyDomain\TestUser"


' Use the NameTranslate object to convert the name to the Distinguished Name.
Set objTrans = CreateObject("NameTranslate")


objTrans.Init ADS_NAME_INITTYPE_GC, ""
objTrans.Set ADS_NAME_TYPE_NT4, strNTName
strUserDN = objTrans.Get(ADS_NAME_TYPE_1779)


' Replace any "/" characters with "\/".
' All other characters that need to be escaped already are escaped.
strUserDN = Replace(strUserDN, "/", "\/")


' Bind to the user object in Active Directory.
Set objUser = GetObject("LDAP://" & strUserDN)
```

The same thing happens if you use ADO to retrieve the value of the distinguished name attribute. All characters will be properly escaped except for any `"/"` characters. For example:

```
' Setup ADO objects.
Set adoCommand = CreateObject("ADODB.Command")
Set adoConnection = CreateObject("ADODB.Connection")
adoConnection.Provider = "ADsDSOObject"
adoConnection.Open "Active Directory Provider"
Set adoCommand.ActiveConnection = adoConnection


' Search entire Active Directory domain.
Set objRootDSE = GetObject("LDAP://RootDSE")
strDNSDomain = objRootDSE.Get("defaultNamingContext")
strBase = "<LDAP://" & strDNSDomain & ">"


' Filter on user objects.
strFilter = "(&(objectCategory=person)(objectClass=user))"


' Comma delimited list of attribute values to retrieve.
strAttributes = "distinguishedName"


' Construct the LDAP syntax query.
strQuery = strBase & ";" & strFilter & ";" & strAttributes & ";subtree"
adoCommand.CommandText = strQuery
adoCommand.Properties("Page Size") = 200
adoCommand.Properties("Timeout") = 30
adoCommand.Properties("Cache Results") = False


' Run the query.
Set adoRecordset = adoCommand.Execute


' Enumerate the resulting recordset.
Do Until adoRecordset.EOF
    ' Retrieve values.
    strDN = adoRecordset.Fields("distinguishedName").value
    ' Replace any "/" characters with "\/".
    ' All other characters that need to be escaped already are escaped.
    strDN = Replace(strDN, "/">, "\/")
    ' Bind to user object.
    Set objUser = GetObject("LDAP://" & strDN)
```

```
    Wscript.Echo "NT Name: " & objUser.sAMAccountName _
        & ", First Name: " & objUser.givenName _
        & ", Last Name: " & objUser.sn
    ' Move to the next record in the recordset.
    adoRecordset.MoveNext
Loop


' Clean up.
adoRecordset.Close
adoConnection.Close
```

↑ Return to Top

PowerShell

The situation is different if you use PowerShell. You still must escape most of the characters required by Active Directory, using the backslash "\" escape character, if they appear in distinguished names. However, PowerShell also requires that the backtick "`" and dollar sign "$" characters be escaped if they appear in any string that is quoted with double quote characters. The backtick is also called the back apostrophe. If the string is quoted with single quote characters, then the backtick is taken literally and cannot be used to escape characters. However, in single quoted strings the "$" characters is also taken literally and does not need to be escaped. The PowerShell escape character is the backtick "`" character. This applies whether you are running PowerShell statements interactively, or running PowerShell scripts.

For unknown reasons, the pound sign "#" does not need to be escaped as part of a distinguished name in PowerShell. This is despite the fact that when PowerShell retrieves a distinguished name that includes the "#" character, it is escaped with the backslash character. Also, the dollar sign "$" need not be escaped if it is the last character in a PowerShell string, word, or token. Of course, it never hurts to escape any character.

If you use the [ADSI] or [ADSISearcher] accelerators (or the equivalent [System.DirectoryServices.DirectoryEntry] or [System.DirectoryServices.DirectorySearcher] classes), or if you use the ADO COM object in PowerShell, the forward slash character "/" must be escaped with the backslash "\" escape character in distinguished names. These technologies use ADSI. But if you use the new Active Directory cmdlets available with Windows Server 2008 R2, like Get-ADUser, the forward slash "/" does not need to be escaped. The new AD modules do not use ADSI.

Finally, if your PowerShell strings are quoted with double quotes, then any double quote characters in the string must be escaped with the backtick "`" character. Alternatively, the embedded double quote characters can be doubled (replace each " character with ""). Any single quote characters would not need to be escaped. Of course, the situation is reversed if the PowerShell string is quoted with single quotes. In that case, single quote characters cannot be escaped with the backtick "`", so you must double the embedded single quotes (replace each ' with ''). Double quote characters would not need to be escaped in a single quoted string. The single quote (') character does not need to be escaped in Active Directory distinguished names, but the double quote (") character does. This means that if you hard code a distinguished name in PowerShell, and the string is enclosed in double quotes, any embedded double quotes must be escaped first by a backtick "`", and then by a backslash "\". A few examples should clarify the situation. Below are some Relative Distinguished Names as they might appear in Active Directory Users and Computers, and how they must be hard coded as part of a distinguished name in PowerShell:

Relative Distinguished Name	Distinguished Name
James "Jim" Smith	"cn=James \`"Jim\`" Smith,ou=West,dc=MyDomain,dc=com"
James $mith	"cn=James `$mith,ou=West,dc=MyDomain,dc=com"
James $mith	'cn=James $mith,ou=West,dc=MyDomain,dc=com'
Sally Wilson + Jones	"cn=Sally Wilson \+ Jones,ou=West,dc=MyDomain,dc=com"
William O'Brian	"cn=William O'Brian,ou=West,dc=MyDomain,dc=com"
William O'Brian	'cn=William O''Brian,ou=West,dc=MyDomain,dc=com'
William O`Brian	"cn=William O``Brian,ou=West,dc=MyDomain,dc=com"
Richard #West	"cn=Richard #West,ou=West,dc=MyDomain,dc=com"
Roy Johnson$	"cn=Roy Johnson$,ou=West,dc=MyDomain,dc=com"

Sam$ Williams	"cn=Sam$ Williams,ou=West,dc=MyDomain,dc=com"

Note the instances of " are replaced with \`", while $ and ` characters are both escaped with the backtick (because it is required in PowerShell) in strings quoted with double quotes. The $ character need not be escaped if the string is quoted with single quotes. Also, the # character does not need to be escaped in PowerShell, and the $ character need not be escaped if is the trailing character in any string, word, or token.

As noted before, unless you use the Active Directory modules, PowerShell depends on ADSI. If any distinguished names are retrieved in PowerShell scripts, you will need to escape any embedded forward slash characters before using the values to bind to the Active Directory object. In the following example, the user with sAMAccountName equal to "Win2kXP" has distinguished name "cn=Windows 2000/XP User,ou=West,dc=MyDomain,dc=com". If the step that replaces "/" with "\/" in the script below is omitted, no error will be raised. However, the object $User will have no properties because there is no such user in Active Directory, so the last statement will not output any value:

```
$Name = "Win2kXP"
$Searcher =  [ADSISearcher]"(sAMAccountName=$Name)"
$Result =  $Searcher.FindOne()
$DN =  $Result.Properties.Item("distinguishedName")
$DN = $DN[0].Replace("/", "\/")
$User =  [ADSI]"LDAP://$DN"
$User.displayName
```

If, however, you use the Path property of the $Result object to retrieve the ADsPath of the user, the resulting value will have the forward slash character properly escaped. For example, this script retrieving values for the same user displays the proper value of the displayName attribute:

```
$Name = "Win2kXP"
$Searcher =  [ADSISearcher]"(sAMAccountName=$Name)"
$Result =  $Searcher.FindOne()
$Path =  $Result.Path
$User =  [ADSI]"$Path"
$User.displayName
```

The first script retrieves the value of the distinguishedName attribute directly from Active Directory, and this value does not have the forward slash character escaped. But the Path property is actually a method that calculates the ADsPath of the object in Active Directory, and whomever coded this method thought to escape the forward slash character. However, this actually could lead to unintended consequences if the situation is not fully understood. The following script may be a bit concocted, but an error is raised if you remove the statement that replaces "\/" with "/", assuming the same user as before:

```
$Name = "Win2kXP"
$Searcher =  [ADSISearcher]"(sAMAccountName=$Name)"
$Result =  $Searcher.FindOne()
$Path =  $Result.Path
$Path =  $Path.Replace("LDAP://", "")
$Path = $Path.Replace("\/", "/")
$User = Get-ADUser $Path -Properties displayName
$User.displayName
```

↑ Return to Top

PowerShell New-AD* Cmdlets

The PowerShell AD Module cmdlets to create new objects, like New-ADUser, New-ADComputer, New-ADGroup, New-ADObject, and New-ADOrganizationalUnit, will automatically escape any characters required in Active Directory. However, if you use any of the Get-AD* cmdlets to retrieve information from Active Directory, and you specify a distinguished name, all characters must be properly escaped.

↑ Return to Top

ADsPath

The ADsPath of any Active Directory object is the distinguishedName with the moniker "LDAP://" appended to the beginning. The ADsPath is not saved in Active Directory, but there is an ADsPath method (also the Path method in PowerShell) which returns the value based on the distinguishedName. The ADsPath retrieved using ADSI will have all characters properly escaped, including the forward slash. This is true whether you use VBScript or PowerShell.

↑ Return to Top

LDAP Filters

The LDAP filter specification assigns special meaning to the following characters:

```
* ( ) \ NUL
```

The `NUL` character is ASCII 00. For example, if you use ADO to query Active Directory, and you use the LDAP syntax, one of the clauses in the query is a filter clause. Command line utilities like adfind and dsquery also accept LDAP filters. Many PowerShell AD cmdlets also accept LDAP filters. In LDAP filters these 5 characters should be escaped with the backslash escape character, followed by the two character ASCII hexadecimal representation of the character. The following table documents this:

Character	Hex Representation
*	\2A
(	\28
)	\29
\	\5C
Nul	\00

For example, if you use ADO in a VBScript program to query for the user with Common Name "`James Jim*) Smith`", the LDAP filter would be:

```
strFilter = "(cn=James Jim\2A\29 Smith)"
```

Actually, the parentheses only need to be escaped if they are unmatched, as above. If instead the Common Name were "`James (Jim) Smith`", nothing would need to be escaped. However, any characters, including non-display characters, can be escaped in a similar manner in an LDAP filter.

↑ Return to Top

Command Line Utilities

Command line utilities like dsquery, dsget, or Joe Richards' free adfind do not use ADSI. Only the ten characters in the first table on this page must be escaped in distinguished names.

The dsget command requires that comma, backslash, and quote characters be escaped. However, the dsquery command only escapes the comma and backslash characters. The quote character must be escaped because the distinguished names output by dsquery are enclosed in quotes, in case there are spaces. The dsget command is fooled by any quote characters embedded in the distinguished names that dsquery fails to escape. For example, the following command at the command prompt of a domain controller should output the sAMAccountName of all users in the domain.

```
dsquery user -limit 0 | dsget user -samid
```

The command raises an error if any users have a quote in their common name. Hopefully, this should be very rare. A workaround is to redirect the output of the dsquery command to a text file, modify the text file to escape any quote characters with the backslash character, then feed the modified text file to the dsget command. For example, create the text file of user distinguished names with this command:

```
dsquery user -limit 0 > users.txt
```

It is not easy to modify the file users.txt since all lines begin and end with quote characters. You need to replace all other instances of `"` with `\"`. You can use the following command to find all objects where the Relative Distinguished Name includes a quote character (`\22` is the hex equivalent of the quote character):

```
dsquery * -filter (name=*\22*)
```

After modifying the text file to escape the quote characters, use this command:

```
type users.txt | dsget user -samid
```

Unfortunately, the dsquery command fails to properly escape leading spaces in the components of distinguished names. Trailing spaces are properly escaped. The dsget command will actually raise an error if any leading quotes in the distinguished names passed to it are escaped. This makes the two utilities consistent, but confusing. If you pass a distinguished name read from a text file with a leading space properly escaped, dsget will raise an error. If you use dsquery to generate a text file of distinguished names, and any of the values include leading spaces, then an error can be raised if this text file is used in a VBScript or PowerShell program that expects the distinguished names to be properly escaped.

↑ Return to Top

Other Programming Languages

Other programming languages have other rules for escaping certain characters. For example, in C# the backslash character must be escaped. The escape character in C# is the backslash character. The following example demonstrates how to escape a comma in an Active Directory Relative Distinguished Name. You must replace `","` with `"\\,"`.

```
rdn = rdn.Replace(",", "\\,")
```

↑ Return to Top

See Also

- VBA & VBS Portal
- Active Directory: Characters to Escape
- Active Directory: Leading Spaces in Names
- Wiki: Active Directory Domain Services (AD DS) Portal
- Wiki: Portal of TechNet Wiki Portals

↑ Return to Top

Other Resources

This article is based on the following: Characters to Escape

Other references:

- Search Filter Syntax
- Describing the ADSI Path
- Creating a Query Filter
- about_Escape_Characters (PowerShell)
- Escape Sequences (in C)