Active Directory: LDAP Syntax Filters

LDAP syntax filters can be used in many situations to query Active Directory. They can be used in VBScript and PowerShell scripts. Many utilities, like adfind and dsquery *, accept LDAP filters. Many PowerShell Active Directory module cmdlets, like Get-ADUser, Get-ADGroup, Get-ADComputer, and Get-ADObject, accept LDAP filters with the LDAPFilter parameter.

Table of Contents

LDAP Clauses

A filter specifies the conditions that must be met for a record to be included in the recordset (or collection) that results from a query. An LDAP filter has one or more clauses, each enclosed in parentheses. Each clause evaluates to either True or False. An LDAP syntax filter clause is in the following form:

```
(<AD Attribute><comparison operator><value>)
```

The `<AD Attribute>` must the the LDAP Display name of an Active Directory attribute. The allowed comparison operators are as follows:

Operator	Meaning
=	Equality
>=	Greater than or equal to (lexicographical)
<=	Less than or equal to (lexicographical)

Note that the operators "<" and ">" are not supported. Another operator, ~= (which means approximately equal to) is supported, but no case has been found where this is useful in Active Directory. The `<value>` in a clause will be the actual value of the Active Directory attribute. The value is not case sensitive and should not be quoted. The wildcard character "*" is allowed, except when the `<AD Attribute>` is a DN attribute. Examples of DN attributes are distinguishedName, manager, directReports, member, and memberOf. If the attribute is DN, then only the equality operator is allowed and you must specify the full distinguished name for the value (or the "*" character for all objects with any value for the attribute). Do not enclose the DN value in parentheses (as is done erroneously in some documentation). If the attribute is multi-valued, then the condition is met if any of the values in the attribute match the filter. An example LDAP syntax filter clause is:

```
(cn=Jim Smith)
```

This filters on all objects where the value of the cn attribute (the common name of the object) is equal to the string "Jim Smith" (not case sensitive). Filter clauses can be combined using the following operators:

Operator	Meaning
&	AND, all conditions must be met
|	OR, any of the conditions must be met
!	NOT, the clause must evaluate to False

For example, the following specifies that either the cn attribute must be "Jim Smith", or the givenName attribute must be "Jim" and the sn attribute must be "Smith":

```
(|(cn=Jim Smith)(&(givenName=Jim)(sn=Smith)))
```

Conditions can be nested with parentheses, but make sure the parentheses match up.

↑ Return to Top

Special Characters

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

```
* ( ) \ NUL
```

The NUL character is ASCII 00. 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, to find all objects where the common name is "James Jim*) Smith", the LDAP filter would be:

```
(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 and foreign characters, can be escaped in a similar manner in an LDAP filter. For example, here are a few foreign characters:

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

↑ Return to Top

Filter on objectCategory and objectClass

When your filter clause includes the objectCategory attribute, LDAP does some magic to convert the values for your convenience. The objectCategory attribute is a DN attribute. A typical value for an object in Active Directory might be "cn=person,cn=Schema,cn=Configuration,dc=MyDomain,dc=com". You can use a filter clause similar to the following:

```
(objectCategory=cn=person,cn=Schema,cn=Configuration,dc=MyDomain,dc=com)
```

However, Active Directory allows you to instead use the following shortcut:

```
(objectCategory=person)
```

The following table documents the result of various combinations of clauses specifying values for objectCategory and objectClass:

objectCategory	objectClass	Result
person	user	user objects
person		user and contact objects
person	contact	contact objects
	user	user and computer objects
computer		computer objects
user		user and contact objects
	contact	contact objects
	computer	computer objects
	person	user, computer, and contact objects
contact		user and contact objects
group		group objects
	group	group objects
person	organizationalPerson	user and contact objects
	organizationalPerson	user, computer, and contact objects
organizationalPerson		user and contact objects

Use the filter that makes your intent most clear. Also, if you have a choice between using objectCategory and objectClass, it is recommended that you use objectCategory. That is because objectCategory is both single valued and indexed, while objectClass is multi-valued and not indexed (except on Windows Server 2008 and above). A query using a filter with objectCategory will be more efficient than a similar filter with objectClass. Windows Server 2008 domain controllers (and above) have a special behavior that indexes the objectClass attribute. You can take advantage of this if all of your domain controllers are Windows Server 2008, or if you specify a Windows Server 2008 domain controller in your query.

↑ Return to Top

Examples

The following table shows many example LDAP filters that can be useful when you query Active Directory:

Query	LDAP Filter
All user objects	(&(objectCategory=person)(objectClass=user))
All user objects (Note 1)	(sAMAccountType=805306368)
All computer objects	(objectCategory=computer)
All contact objects	(objectClass=contact)
All group objects	(objectCategory=group)
All organizational unit objects	(objectCategory=organizationalUnit)
All container objects	(objectCategory=container)
All builtin container objects	(objectCategory=builtinDomain)
All domain objects	(objectCategory=domain)
Computer objects with no description	(&(objectCategory=computer)(!(description=*)))
Group objects with a description	(&(objectCategory=group)(description=*))
Users with cn starting with "Joe"	(&(objectCategory=person)(objectClass=user) (cn=Joe*))
Object with description "East\West Sales" (Note 2)	(description=East\5CWest Sales)
Phone numbers in form (xxx) xxx-xxx	(telephoneNumber=(*)*-*)
Groups with cn starting with "Test" or "Admin"	(&(objectCategory=group) (|(cn=Test*)(cn=Admin*)))
All users with both a first and last name.	(&(objectCategory=person)(objectClass=user) (givenName=*)(sn=*))

All users with direct reports but no manager	(&(objectCategory=person)(objectClass=user) (directReports=*)(!(manager=*)))
All users with specified email address	(&(objectCategory=person)(objectClass=user) (|(proxyAddresses=*:jsmith@company.com) (mail=jsmith@company.com)))
Object with Common Name "Jim * Smith" (Notes 3, 19)	(cn=Jim \2A Smith)
Objects with sAMAccountName that begins with "x", "y", or "z"	(sAMAccountName>=x)
Objects with sAMAccountName that begins with "a" or any number or symbol except "$"	(&(sAMAccountName<=a)(!(sAMAccountName=$*)))
All users with "Password Never Expires" set (Note 4)	(&(objectCategory=person)(objectClass=user) (userAccountControl:1.2.840.113556.1.4.803:=65536))
All disabled user objects (Note 4)	(&(objectCategory=person)(objectClass=user) (userAccountControl:1.2.840.113556.1.4.803:=2))
All enabled user objects (Note 4)	(&(objectCategory=person)(objectClass=user) (!(userAccountControl:1.2.840.113556.1.4.803:=2)))
All users not required to have a password (Note 4)	(&(objectCategory=person)(objectClass=user) (userAccountControl:1.2.840.113556.1.4.803:=32))
All users with "Do not require kerberos preauthentication" enabled	(&(objectCategory=person)(objectClass=user) (userAccountControl:1.2.840.113556.1.4.803:=4194304))
Users with accounts that do not expire (Note 5)	(&(objectCategory=person)(objectClass=user) (|(accountExpires=0) (accountExpires=9223372036854775807)))
Users with accounts that do expire (Note 5)	(&(objectCategory=person)(objectClass=user) (accountExpires>=1) (accountExpires<=9223372036854775806))
Accounts trusted for delegation (unconstrained delegation)	(userAccountControl:1.2.840.113556.1.4.803:=524288)
Accounts that are sensitive and not trusted for delegation	(userAccountControl:1.2.840.113556.1.4.803:=1048574)
All distribution groups (Notes 4, 15)	(&(objectCategory=group) (!(groupType:1.2.840.113556.1.4.803:=2147483648)))
All security groups (Notes 4, 19)	(groupType:1.2.840.113556.1.4.803:=2147483648)
All built-in groups (Notes 4, 16, 19)	(groupType:1.2.840.113556.1.4.803:=1)
All global groups (Notes 4, 19)	(groupType:1.2.840.113556.1.4.803:=2)
All domain local groups (Notes 4, 19)	(groupType:1.2.840.113556.1.4.803:=4)
All universal groups (Notes 4, 19)	(groupType:1.2.840.113556.1.4.803:=8)
All global security groups (Notes 17, 19)	(groupType=-2147483646)
All universal security groups (Notes 17, 19)	(groupType=-2147483640)
All domain local security groups (Notes 17, 19)	(groupType=-2147483644)
All global distribution groups (Note 19)	(groupType=2)
All objects with service principal name	(servicePrincipalName=*)
Users with "Allow Access" on "Dial-in" tab of ADUC (Note 6)	(&(objectCategory=person)(objectClass=user) (msNPAllowDialin=TRUE))
Users with "Control access though NPS Network Policy" on "Dial-in" tab of ADUC	(&(objectCategory=person)(objectClass=user) (!(msNPAllowDialin=*)))
All groups created after March 1, 2011	(&(objectCategory=group) (whenCreated>=20110301000000.0Z))
All users that must change their password at next logon	(&(objectCategory=person)(objectClass=user) (pwdLastSet=0))
All users that changed their password since April 15, 2011 (CST) (Note 7)	(&(objectCategory=person)(objectClass=user) (pwdLastSet>=129473172000000000))
All users with "primary" group other than "Domain Users"	(&(objectCategory=person)(objectClass=user) (!(primaryGroupID=513)))
All computers with "primary" group "Domain Computers"	(&(objectCategory=computer) (primaryGroupID=515))
Object with GUID "90395F191AB51B4A9E9686C66CB18D11" (Note 8)	(objectGUID=\90\39\5F\19\1A\B5\1B\4A\9E\96 \86\C6\6C\B1\8D\11)
Object beginning with GUID "90395F191AB51B4A" (Note 8)	(objectGUID=\90\39\5F\19\1A\B5\1B\4A*)
Object with SID "S-1-5-21-73586283 -152049171-839522115-1111" (Note 9)	(objectSID=S-1-5-21-73586283-152049171 -839522115-1111)
Object with SID "0105000000000005150000006BD662041316100943170A3257040000"	(objectSID=\01\05\00\00\00\00\00\05\15 \00\00\00\6B\D6\62\04\13\16\10\09\43\17\0A\32

(Note 9)	\57\04\00\00)
All computers that are not Domain Controllers (Note 4)	(&(objectCategory=computer) (!(userAccountControl:1.2.840.113556.1.4.803:=8192)))
All Domain Controllers (Note 4)	(&(objectCategory=computer) (userAccountControl:1.2.840.113556.1.4.803:=8192))
All Domain Controllers (Notes 14, 19)	(primaryGroupID=516)
All servers	(&(objectCategory=computer) (operatingSystem=*server*))
All member servers (not DC's) (Note 4)	(&(objectCategory=computer) (operatingSystem=*server*) (!(userAccountControl:1.2.840.113556.1.4.803:=8192)))
All direct members of specified group	(memberOf=cn=Test,ou=East,dc=Domain,dc=com)
All users not direct members of a specified group	(&(objectCategory=person)(objectClass=user) (!(memberOf=cn=Test,ou=East,dc=Domain,dc=com)))
All groups with specified direct member (Note 19)	(member=cn=Jim Smith,ou=West, dc=Domain,dc=com)
All members of specified group, including due to group nesting (Note 10)	(memberOf:1.2.840.113556.1.4.1941:= cn=Test,ou=East,dc=Domain,dc=com)
All groups specified user belongs to, including due to group nesting (Notes 10, 19)	(member:1.2.840.113556.1.4.1941:= cn=Jim Smith,ou=West,dc=Domain,dc=com)
Objects with givenName "Jim*" and sn "Smith*", or with cn "Jim Smith*" (Note 11)	(anr=Jim Smith)
All attributes in the Schema container replicated to the GC (Notes 6, 12)	(&(objectCategory=attributeSchema) (isMemberOfPartialAttributeSet=TRUE))
All operational (constructed) attributes in the Schema container (Notes 4, 12)	(&(objectCategory=attributeSchema) (systemFlags:1.2.840.113556.1.4.803:=4))
All attributes in the Schema container not replicated to other Domain Controllers (Notes 4, 12)	(&(objectCategory=attributeSchema) (systemFlags:1.2.840.113556.1.4.803:=1))
All objects where deletion is not allowed (Notes 4)	(systemFlags:1.2.840.113556.1.4.803:=2147483648)
Attributes whose values are copied when the object is copied (Notes 4, 12)	(searchFlags:1.2.840.113556.1.4.803:=16)
Attributes preserved in tombstone object when object deleted (Notes 4, 12)	(searchFlags:1.2.840.113556.1.4.803:=8)
Attributes in the Ambiguous Name Resolution (ANR) set (Notes 4, 12)	(searchFlags:1.2.840.113556.1.4.803:=4)
Attributes in the Schema that are indexed (Notes 4, 12)	(searchFlags:1.2.840.113556.1.4.803:=1)
Attributes marked confidential in the schema (Notes 4, 12)	(searchFlags:1.2.840.113556.1.4.803:=128)
Attributes in the RODC filtered attribute set, or FAC (Notes 4, 12)	(searchFlags:1.2.840.113556.1.4.803:=512)
All site links in the Configuration container (Note 13)	(objectClass=siteLink)
The nTDSDSA objects associated with all Global Catalogs. This will identify all DC's that are GC's. (Note 4)	(&(objectCategory=nTDSDSA) (options:1.2.840.113556.1.4.803:=1))
The nTDSDSA object associated with the PDC Emulator. This will identify the DC with the PDC Emulator FSMO role (Note 18).	(&(objectClass=domainDNS)(fSMORoleOwner=*))
The nTDSDSA object associated with the RID Master. This will identify the DC with the RID Master FSMO role (Note 18).	(&(objectClass=rIDManager)(fSMORoleOwner=*))
The nTDSDSA object associated with the Infrastructure Master. This will identify the DC with this FSMO role (Note 18).	(&(objectClass=infrastructureUpdate) (fSMORoleOwner=*))
The nTDSDSA object associated with the Schema Master. This will identify the DC with the Schema Master FSMO role (Note 18).	(&(objectClass=dMD)(fSMORoleOwner=*))
The nTDSDSA object associated with the Domain Naming Master. This will identify the DC with this FSMO role (Note 18).	(&(objectClass=crossRefContainer) (fSMORoleOwner=*))
All Exchange servers in the Configuration container (Note 13)	(objectCategory=msExchExchangeServer)
All objects protected by AdminSDHolder	(adminCount=1)
All trusts established with a domain	(objectClass=trustedDomain)
All Group Policy objects	(objectCategory=groupPolicyContainer)
All service connection point objects	(objectClass=serviceConnectionPoint)
All Read-Only Domain Controllers (Notes 4, 19)	(userAccountControl:1.2.840.113556.1.4.803:=67108864)

NOTES

1. The filter `(sAMAccountType=805306368)` for user objects is more efficient than the more usual filter, but is harder to remember.
2. The backslash character must be escaped in LDAP filters. Substitute `\5C`.
3. The asterisk character must be escaped in LDAP filters. Substitute `\2A`.
4. The string `1.2.840.113556.1.4.803` specifies `LDAP_MATCHING_RULE_BIT_AND`. This specifies a bitwise AND of a flag attribute (an integer), like userAccountControl, groupType, or systemFlags, and a bit mask (like 2, 32, or 65536). The clause is True if the bitwise AND of the attribute value and the bit mask is non-zero, indicating the bit is set.
5. The accountExpires attribute is Integer8, a large 64-bit integer representing a date (in UTC) as the number of 100-nanosecond intervals since 12:00 AM January 1, 1601. If an account does not expire, then accountExpires is either 0 or 2^63-1 (9,223,372,036,854,775,807 the largest 64-bit integer allowed), both of which mean never.
6. To filter on Boolean Active Directory attributes, like msNPAllowDialin or isMemberOfPartialAttributeSet, make sure the values `TRUE` or `FALSE` are all uppercase in the clause. This is the only time comparisons are case sensitive.
7. The pwdLastSet attribute is Integer8.
8. Byte arrays, like the objectGUID attribute, can be represented as a series of escaped hexadecimal bytes. The GUID `{b95f3990-b59a-4a1b-9e96-86c66cb18d99}` is equivalent to the hex representation "`90395fb99ab51b4a9e9686c66cb18d99`". Notice how the order of the first 8 bytes is reversed in groups. You specify the escaped hex bytes. You cannot specify the form in curly braces in a filter.
9. The objectSID attribute is saved in Active Directory as a byte array. You can either specify the decimal display format `S-1-5-21-73586283-152049171-839522115-1111` or the equivalent hex representation where each byte is escaped "`\01\05\00\00\00\00\00\05\15\00\00\00\6B\D6\62\04\13\16\10\09\43\17\0A\32\57\04\00\00`". The later might be easier in VBScript.
10. The string `1.2.840.113556.1.4.1941` specifies `LDAP_MATCHING_RULE_IN_CHAIN`. This applies only to DN attributes. This is an extended match operator that walks the chain of ancestry in objects all the way to the root until it finds a match. This reveals group nesting. It is available only on domain controllers with Windows Server 2003 SP2 or Windows Server 2008 (or above).
11. The string "`anr`" is an acronym for "Ambiguous Name Resolution". See the link below for complete explanation.
12. To query for attributes in the Schema, the base of the query must be the Schema container, such as `cn=Schema,cn=Configuration,dc=MyDomain,dc=com`.
13. To query for objects in the Configuration container the base of the query must be the Configuration container, such as `cn=Configuration,dc=MyDomain,dc=com`.
14. The "primary" group for all Domain Controllers should be the group "Domain Controllers", which has the well-known RID 516.
15. Many LDAP filters for various types of Active Directory groups can use the groupType attribute and skip the usual (objectCategory=group) clause. This is because only group objects can have the groupType attribute. For example, the filter (groupType=2) will retrieve all global distribution groups. However, if the filter uses the "Not" operator, such as (!(groupType:1.2.840.113556.1.4.803:=2147483648)) for all distribution groups (groups that are not security groups), you will also retrieve all objects that do not have the groupType attribute. In this case you must "And" this clause with the (objectCategory=group) clause.
16. You might expect the LDAP filter for built-in security groups to be (groupType=2147483649) or (groupType=-2147483643). This is because the bit-wise "Or" of 2,147,483,648 (the bit mask for security groups) and 1 (the bit mask for built-in groups) would result in these values. However, this returns no results. The reason is that the built-in groups in Active Directory are also domain local. You need to account for this by Or'ing these values with 4, the bit mask for domain local groups. The result is (2,147,483,643 Or 1 Or 4) = 2,147,483,653, which after subtracting 2^32 (see Note 17) becomes -2,147,483,643. You can use either (groupType=2147483653) or (groupType=-2147483643) to retrieve all built-in domain local security groups. However, it probably makes more sense to just filter on all built-in groups with (groupType:1.2.840.113556.1.4.803:=1).
17. The userAccountControl and groupType attributes in Active Directory are 32-bit integers. This means the values can range from -2^31 to 2^31 - 1, or -2,147,483,648 to 2,147,483,647 (the commas are shown here for readability, but are not allowed in filters). The values assigned to these attributes will be the result of a bit-wise "Or" of the appropriate bit mask for each setting. For example, the value assigned to the groupType attribute of a universal security group will be the "Or" of the bit mask for a universal group, which is 8, and the bit mask for a security group, which is 2,147,483,648. The result of (8 Or 2,147,483,648) is 2,147,483,656. Technically this value is not possible as it exceeds the maximum allowed for a 32-bit integer. Instead, the system "wraps" the value into a negative number. The value 2,147,483,656 becomes -2,147,483,640. The rule is that if the value of a 32-bit integer is larger than 2^31 -1, subtract 2^32 (which is 4,294,967,296). The value of the groupType attribute for a universal security group becomes 2,147,967,296 - 4,294,967,296 = -2,147,483,640. This is the value your will see in Active Directory using ADSI Edit. Most utilities, scripts, and programs that accept LDAP syntax filters will work correctly with either value. However, in case the utility can only handle 32-bit integers it would be safest to use the negative number. Also, the VBScript bit-wise operators (And, Or, Xor, Not) can only handle 32-bit integers.
18. There are five FSMO roles. For the PDC Emulator, RID Master, and Infrastructure Master roles the base of the query should be the domain. There is one of these FSMO roles for each domain. There is one Schema Master and one Domain Naming Master role for the forest. The base of the query to search for the Schema Master role should be the schema container, such as cn=Schema,cn=Configuration,dc=MyDomain,dc=com. The base of the query for the Domain Naming Master role should be the Configuration container, such as cn=Configuration,dc=MyDomain,dc=com. In all cases, the query will retrieve a nTDSDSA object. The parent of this object will have a Relative Distinguished Name identical to that of the corresponding DC. This parent object has a dnsHostName attribute whose value is the DNS name of the DC with the FSMO role.
19. Many times you can take advantage of the fact that only one class of object in Active Directory has a particular attribute. For example, only group objects have the groupType and member attributes. This allows you to filter on groupType with a clause like (groupType=2) without using a second clause restricting the query to group objects, like (objectCategory=group). However, if your query only has the one filter, it will be checked against all objects in Active Directory. It turns out that if you also use the second clause (to restrict the query to groups), it runs faster. The results will be the same, and in most cases the time difference doesn't matter much, but a filter like (& (objectCategory=group)(member=cn=Jim Smith,ou=West,dc=MyDomain,dc=com)) is much faster than simply (member=cn=Jim Smith,ou=West,dc=MyDomain,dc=com).
20. According to RFC 2254, the NOT operator, "!", should operate on a clause in parentheses (similar to the operators | and &). Although a clause similar to (!cn=*Smith) works in almost all cases, it would be more correct to use (!(cn=*Smith)). The first form works in VBScript, PowerShell V1, using the -LDAPFilter parameter with the PowerShell AD modules, in dsquery *, and with Joe Richards' adfind utility. However, cases have been reported where it raises an error.

↑ Return to Top

Testing

You can use the following PowerShell V1 script to test various LDAP syntax filters in your environment:

Generic Search of Active Directory: http://gallery.technet.microsoft.com/Generic-Search-of-Active-0a05b8d0

↑ Return to Top

External Links

This article is based on: ADO Search Tips

Other references:

- Search Filter Syntax
- ANR in ADO Searches (Ambiguous Name Resolution)
- How Active Directory Searches Work

↑ Return to Top

See Also

- Active Directory: Ambiguous Name Resolution
- VBA & VBS Portal
- Wiki: Active Directory Domain Services (AD DS) Portal
- Wiki: Portal of TechNet Wiki Portals

↑ Return to Top

Other Languages

This article is also available in the following languages:

Russian (ru-RU)

- Active Directory: Использование LDAP-фильтров (ru-RU)

↑ Return to Top
