Powershell - Search Active Directory using PowerShell ADSISearcher Filters

https://www.alkanesolutions.co.uk/2021/03/03/search-active-directory-using-adsisearcher-filters/

How to Search

This post discusses how we can search Active Directory using PowerShell ADSISearcher filters. Using search filters can improve search performance significantly.

Consider the following where we create a default ADSISearcher to begin searching Active Directory (AD):

```
$objSearcher=[adsisearcher]""
```

If we used this default configuration, the ADSISearcher would search every object in every organisation unit (OU) in AD. We would then need to filter which records we want to process in some kind of iterative loop after the search results have been retrieved. This would be inefficient and extremely slow.

Luckily the ADSISearcher is a type accelerator for System.DirectoryServices.DirectorySearcher, which exposes a property called 'Filter'.

We can use this to filter for only users like so:

```
$objSearcher.Filter = "(objectClass=user)"
```

or perhaps search for all disabled users:

```
$objSearcher.Filter = "(&(objectClass=user)(!userAccountControl:1.2.840.113556.1.4.803:=2))"
```

If we want to search for a specific user:

```
$objSearcher.Filter = "(&(objectClass=user)(sAMAccountName=alkaneuser1))"
```

or maybe we only know part of the username, and need to use the asterisk as a wildcard like so:

```
$objSearcher.Filter = "(&(objectClass=user)(sAMAccountName=*lkaneus*))"
```

What you may notice in the above examples is that we can also filter using AND or OR logical operators.

The syntax for logical operators is an ampersand (&) for AND, and a pipe symbol (|) for OR. And all of the expressions should be encased inside brackets.

Here we search where sAMAccountName is either alkaneuser1 **OR** alkaneuser2:

```
(|(sAMAccountName=alkaneuser1)(sAMAccountName=alkaneuser2))
```

Here we search where sAMAccountName is alkaneuser1 **AND** title is ‘Project Manager’:

```
(&(sAMAccountName=kt04ag)(title=Project Manager))
```

Here’s an example of how we could combine multiple logical operators to search for only user objects **AND** search where sAMAccountName is alkaneuser1 **OR** alkaneuser2:

```
$objSearcher.Filter =
"(&(objectClass=user)(|(sAMAccountName=alkaneuser1)(sAMAccountName=alkaneuser2)))"
```

Of course, it’s not only users we can search for. In this example we filter our search by computer objects only **AND** where the computer is called alkanecomputer1:

```
$objSearcher.Filter = "(&(objectClass=computer)(Name=alkanecomputer1))"
```

The SearchRoot property can also optimise LDAP queries. By specifying this, we’re only searching in a specific OU as opposed to the whole of AD! We can specify this like so:

```
$objSearcher.SearchRoot = [ADSI]"LDAP://OU=Users,DC=alkanesolutions,DC=co,DC=uk"
```

Another important property to mention when searching AD is PageSize. If the data we are returning contains more than 1000 items, we must page the results otherwise you’ll likely run into ‘LDAP_SIZELIMIT_EXCEEDED’ errors. I typically set it to 200 like so:
$objSearcher.PageSize = 200

Another optimisation is to define which LDAP properties should be returned. There are many, many LDAP roperties and if we don’t require them all we can significantly improve the speed of LDAP queries by specifying only the properties we require like so:

```
$colProplist =
"name","givenname","distinguishedname","description","displayname","samaccountname","title","mail","depart
ment"
foreach ($i in $colPropList){$objSearcher.PropertiesToLoad.Add($i) | out-null }
```

Finally when we search AD using LDAP, we should already know if we're expecting many results or just one result. If we're expecting many results we should use FindAll() and iterate through each result like so:

```
$allObjects = $objSearcher.FindAll()
foreach ($obj in $allObjects) {
write-host ($obj.properties).name
write-host ($obj.properties).displayname
write-host ($obj.properties).givenname
write-host ($obj.properties).distinguishedname
write-host ($obj.properties).description
write-host ($obj.properties).samaccountname
write-host ($obj.properties).title
write-host ($obj.properties).mail
write-host ($obj.properties).department
}
```

However if we're only searching for one result (maybe to return information for a single user) we should use FindOne() since it returns the first result only. We can use it like so:

```
$firstObject = $objSearcher.FindOne()
if ($firstObject -ne $null) {
write-host ($firstObject.properties).name
write-host ($firstObject.properties).displayname
write-host ($firstObject.properties).givenname
write-host ($firstObject.properties).distinguishedname
write-host ($firstObject.properties).description
write-host ($firstObject.properties).samaccountname
write-host ($firstObject.properties).title
write-host ($firstObject.properties).mail
write-host ($firstObject.properties).department
}
```

You can play around with the full script here:

```
$objSearcher=[adsisearcher]""
$objSearcher.Filter = "(&(objectClass=user)(sAMAccountName=alkaneuser1))"
```

```
#$objSearcher.Filter = "(&(objectClass=user)(sAMAccountName=*lkaneus*))"
#$objSearcher.Filter =
"(&(objectClass=user)(|(sAMAccountName=alkaneuser1)(sAMAccountName=alkaneuser2)))"
#$objSearcher.Filter = "(&(objectClass=computer)(Name=alkanecomputer1))"
$objSearcher.PageSize = 200
$colProplist =
"name","givenname","distinguishedname","description","displayname","samaccountname","title","mail","depart
ment"
foreach ($i in $colPropList){$objSearcher.PropertiesToLoad.Add($i) | out-null }
$objSearcher.SearchRoot = [ADSI]"LDAP://OU=Users,DC=alkanesolutions,DC=co,DC=uk"
$allObjects = $objSearcher.FindAll()
foreach ($obj in $allObjects) {
write-host ($obj.properties).name
write-host ($obj.properties).displayname
write-host ($obj.properties).givenname
write-host ($obj.properties).distinguishedname
write-host ($obj.properties).description
write-host ($obj.properties).samaccountname
write-host ($obj.properties).title
write-host ($obj.properties).mail
write-host ($obj.properties).department
}
$firstObject = $objSearcher.FindOne()
if ($firstObject -ne $null) {
write-host ($firstObject.properties).name
write-host ($firstObject.properties).displayname
write-host ($firstObject.properties).givenname
write-host ($firstObject.properties).distinguishedname
write-host ($firstObject.properties).description
write-host ($firstObject.properties).samaccountname
write-host ($firstObject.properties).title
write-host ($firstObject.properties).mail
write-host ($firstObject.properties).department
}
```

Revision #1
Created 2 September 2024 20:09:05 by Steve Ling
Updated 2 September 2024 20:11:28 by Steve Ling