



[TechNet Home](#) > [Script Center](#) > [Learn to Script](#)

Tales from the Script

April 2005

Published: April 1, 2005



By [The Scripting Guys](#)

For a list and additional information on all Tales from the Script columns, [click here](#).

On This Page

- ↓ [Dude, Where's My Printer: Using Scripts to Search Active Directory \(Part 1\)](#)
- ↓ [Did You Say Active Directory?](#)
- ↓ [Creating a Connection to Active Directory Using the Active Directory ADO Provider](#)
- ↓ [Specifying Command Object Parameters](#)
- ↓ [Page Size](#)
- ↓ [Searchscope](#)
- ↓ [Sort On](#)
- ↓ [Creating a Query](#)
- ↓ [Executing the Query](#)
- ↓ [Looping Through the Returned Recordset](#)
- ↓ [Stay Tuned for Our Next Exciting Episode](#)



Dude, Where's My Printer: Using Scripts to Search Active Directory (Part 1)

If you've ever spent a Saturday afternoon wandering aimlessly around the house looking for your car keys or the TV remote, don't feel bad: you're in good company. The truth is, human history often seems to be little more than an account of people searching for (and typically not finding) things, be they the lost continent of Atlantis, the Fountain of Youth, or the newspaper that one Microsoft Scripting Guy once lost in the short walk between his driveway and his living room.

Note. Don't ask. Let's just say that, for this Scripting Guy, that's not an uncommon occurrence.

If you're a system administrator, you've probably embarked on a few great expeditions of discovery and

exploration all your own:

"I need to know how many of our computers are running Windows XP, and how many of those have Service Pack 2 installed."

"Don't we already have plenty of color printers in our Detroit office?"

"We need a list of all our employees and their home phone numbers."

Unfortunately, for many of us these quests are as fruitless as the quests to find El Dorado, the fabled City of Gold. Explorers looking for El Dorado have typically just wandered through the jungles of South America, hoping to stumble upon the legendary treasure. So far that approach hasn't worked out real well; in fact, Sir Walter Raleigh was executed, in part due to his failure to find El Dorado. (Fortunately at Microsoft we don't execute people just for making a mistake. Although, come to think of it, anyone seen [Peter](#) around lately?)

Likewise, those system administrators (or their surrogates) at the Detroit office will typically wander through cubicles hoping to stumble upon a color printer. And while this approach has yet to turn up the city of El Dorado, it usually *will* turn up all the color printers in the Detroit office; it's just that this process takes a lot of time, can cost a lot of money (after all, the money being paid you or your surrogate to wander the hallways in Detroit could have been spent on other projects), and is highly-susceptible to errors. Isn't there a better way?

As a matter of fact there is: all you have to do is write a simple little script that searches Active Directory.

[↑ Top of page](#)

Did You Say Active Directory?

Yes, Active Directory. Everyone knows that Active Directory is the place where user and computer accounts are stored, but not everyone realizes how much other important information is (or at least can be) stored in Active Directory. Some of this information comes for free. For example, when you join a computer to the domain, information such as the operating system installed on that computer (and the service pack version as well) is automatically added to the computer account; when you publish a printer in Active Directory you get such additional information as whether the printer supports color printing and whether it supports stapling and duplexing.

Other information requires a little more effort on your part: needless to say, when you add a new user to Active Directory, the directory service isn't automatically populated with the user's cell phone number and the name of the department the user works in. However, space *has* been set aside in Active Directory for things like an Employee ID number and a job title. If you take it upon yourself to supply that information, you'll have a rich database that can be mined over and over again using simple scripts.

Note. Just how rich *is* this mother lode? Rich enough that this *Tales from the Script* column is actually Part 1 of a two-part series. This month we'll cover the basics of searching Active Directory; next month we'll talk more about what you can search for and how you can incorporate searches in your system administration scripts. This might also be a good time to tell you that this month's column assumes a basic familiarity with ADSI: Active Directory Service Interfaces. If you don't have this basic familiarity, you might want to check out the [ADSI Primer](#) in the *Microsoft Windows 2000 Scripting Guide*.

For now, let's talk about how to write scripts to mine this mother lode. First let's take a look at a script that searches for all the user accounts in a domain, and then discuss the script and how it works:

```
On Error Resume Next
Const ADS_SCOPE_SUBTREE = 2

Set objConnection = CreateObject("ADODB.Connection")
Set objCommand = CreateObject("ADODB.Command")
objConnection.Provider = "AdsDSOobject"
objConnection.Open "Active Directory Provider"
```

```

Set objCommand.ActiveConnection = objConnection

objCommand.Properties("Page Size") = 1000
objCommand.Properties("Searchscope") = ADS_SCOPE_SUBTREE
objCommand.Properties("Sort On") = "Name"

objCommand.CommandText = _
    "SELECT Name FROM 'LDAP://dc=fabrikam,dc=com' WHERE objectCategory='user'"
Set objRecordSet = objCommand.Execute

objRecordSet.MoveFirst
Do Until objRecordSet.EOF
    Wscript.Echo objRecordSet.Fields("Name").Value
    objRecordSet.MoveNext
Loop

```

Now, we know what you're thinking: "Maybe I'd be better off searching for El Dorado; fighting off piranha and poisonous snakes sounds better than having to deal with things like ADsDSOObject." Relax: this script's bark is much worse than its bite. For the most part, there are only a handful of items you have to worry about when writing an Active Directory search script. The rest is boilerplate text that can remain unchanged regardless of whether you're searching for all the subnets in a domain or for all the contacts who work for the Contoso company.

Yes, we know: according to legend the king of El Dorado awoke each morning, took a dip in the lake, and then rolled around in gold dust. (Pretty much the same morning routine the Scripting Guys go through.) Very few people believe in the legend of El Dorado these days; why, then, should we expect you to believe in the legend of Active Directory, the notion that you can take a basic search script and – with just a couple of minor modifications – use that single script to search for anything in Active Directory. This time, though, the legend is true.

To prove it to all you skeptics and cynics, let's break this script into four pieces:

- Creating a connection to Active Directory using the Active Directory ADO provider
- Specifying Command object parameters
- Creating and executing a query
- Looping through the returned recordset

Master these four pieces and you'll be well on your way to mastering Active Directory searches.

[↑ Top of page](#)

Creating a Connection to Active Directory Using the Active Directory ADO Provider

Creating a connection to Active Directory using the Active Directory ADO (ActiveX Data Objects) provider is pure boilerplate; in fact, you might very well be able to use this exact same code in each of your scripts, completely unchanged:

```

Set objConnection = CreateObject("ADODB.Connection")
Set objCommand = CreateObject("ADODB.Command")
objConnection.Provider = "ADsDSOObject"
objConnection.Open "Active Directory Provider"
Set objCommand.ActiveConnection = objConnection

```

This is the part that scares some people away. But there's really no reason to be afraid: all you're doing here is creating two objects – the Connection object and the Command object – and then using those two objects to connect to the directory service. The Connection object is used to load the ADO Active Directory database provider and to validate user credentials. After a connection is made, you use the Command object to issue commands and to run queries. It takes five lines of code but, like we said, the same five lines of code will work in all your Active Directory search scripts.

A Brief Aside: Using Alternate Credentials

By default, the Active Directory database provider connects to Active Directory using the same security credentials you used to log on. But what if you want to connect to Active Directory using *different* security credentials? For example, suppose you logged in as a regular user, but now you need to connect to Active Directory as an administrator? No problem; just use code similar to this (the code for specifying the alternate credentials is separated from the rest of the script by blank lines):

```
Set objConnection = CreateObject("ADODB.Connection")
Set objCommand = CreateObject("ADODB.Command")
objConnection.Provider = "ADsDSOObject"

objConnection.Properties("User ID") = "Administrator"
objConnection.Properties("Password") = "irte56$#sw"
objConnection.Properties("Encrypt Password") = TRUE
objConnection.Properties("ADSI Flag") = 1

objConnection.Open "Active Directory Provider"
Set objCommand.ActiveConnection = objConnection
```

In case you're wondering what these properties mean, while searching for the TV remote control we happened to stumble upon a table that explains each one:

Property	Description
User ID	This is simply the user account whose security context will be used when performing the search. Names can be formatted in several ways: kenmyer fabrikam\kenmyer kenmyer@fabrikam.com
Password	The password for the user account specified in the User ID property.
Encrypt Password	A Boolean value that specifies whether the password is encrypted. The default is False.
ADSI Flag	A set of flags that specify the binding authentication options. The default is 0. Flags commonly used in connecting to Active Directory include (values are shown in parentheses): ADS_SECURE_AUTHENTICATION (1). Requests secure authentication. ADS_USE_ENCRYPTION (2). Requires ADSI to use encryption for data exchange over the network. ADS_USE_SIGNING (40). Verifies data integrity. The ADS_SECURE_AUTHENTICATION flag must also be set to use signing. ADS_USE_SEALING (80). Encrypts data using Kerberos. The ADS_SECURE_AUTHENTICATION flag must also be set to use sealing.

[↑ Top of page](#)

Specifying Command Object Parameters

After a connection has been made with Active Directory you then use the Command object to issue queries and to set parameters for your search. We'll talk about issuing queries in a minute; for now, let's talk about the three Command object properties you'll need to deal with:

- Page Size
- Searchscope
- Sort On

[↑ Top of page](#)

Page Size

Probably no single item in the history of scripting has been as misunderstood – and has caused as much grief – as page size. Here’s why. By default, any time you write a script that searches Active Directory you get back a maximum of 1,000 items, period. It doesn’t matter whether you have 1,001 user accounts in your domain or 100,001 user accounts: you only get 1,000 of those accounts returned to you.

But wait, don’t leave; there’s a way to work around that 1,000-record limitation. The trick is to specify a Page Size in your script. Suppose you specify a page size of 500 using code like this:

```
objCommand.Properties("Page Size") = 500
```

With the Page Size set, your script will return the first 500 records it finds. The script will pause momentarily (typically you won’t even notice this pause) and then return the next 500 records it finds. This process will continue until *all* the records have been returned. And, again, it doesn’t matter whether you have 1,001 or 100,001 accounts; eventually, you’ll get them all. *That’s* how you get around the 1,000-record limitation.

Note. Does it make any difference what value you set the Page Size to? In testing on an admittedly small network, we couldn’t see any difference. Depending on your network traffic and the stress and strain placed on your domain controller you might find it better to use a larger or small page size. (The maximum value, by the way, is 1000.) That’s something you’ll have to determine for yourself.

[↑ Top of page](#)

Searchscope

If you’re in the habit of losing things you inevitably develop strategies to make searching easier. One of the most important of these strategies is figuring out where to start your search in the first place. For example, suppose you lose the television remote control. If you’ve never been to North Dakota then it’s probably not the best strategy to immediately fly to North Dakota and start looking for the remote there. (Although there might be some value in flying off to Tahiti and starting the search *there*.) Instead, you should probably identify the most likely place where the remote would be and begin your search at that point.

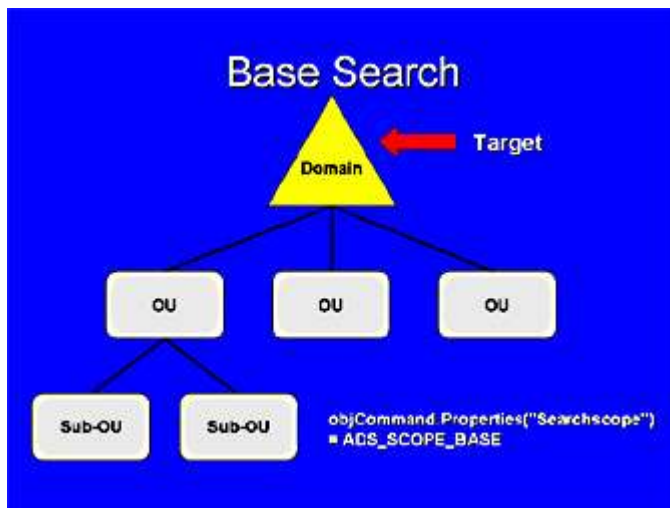
Suppose you had the remote control earlier in the day. Since then you haven’t set foot out of the house. In that case, it’s a pretty good bet that the remote is somewhere in the house; therefore, you might want to constrain your search just to the house. (Unless you have a teenaged son, of course. In that case, it’s a pretty good bet that *he* has the remote. If you don’t have a teenaged son, your best bet is the refrigerator.)

In other words, determining a starting point and identifying search constraints are all part of setting a search scope. And guess what? The strategies that apply when searching for your lost remote control are equally applicable to searching for objects in Active Directory. So let’s take a closer look at the three kinds of search scopes available to script writers.

Base Search

With this type of search you search only the so-called “base” object (that is, the Active Directory container where you begin your search); child containers are *not* searched. For example, if you start your search in the Active Directory root only the root will be searched; no OUs or other containers will be searched. The base search is useful when you want to pull out information for a single OU (for example, a list of all the user accounts in the Finance OU).

Confused? Don’t be. Take a look at the following diagram; the domain container (shown in yellow) is the only item that will be searched when performing a base search that starts in the domain root. That’s because, with a base search, searching is limited to the specified container, in this case, the domain root.



For a full-size version of this graphic, click [here](#).

To specify a base search, do two things. First, create a constant named ADS_SCOPE_BASE and set the value to 0:

```
Const ADS_SCOPE_BASE = 0
```

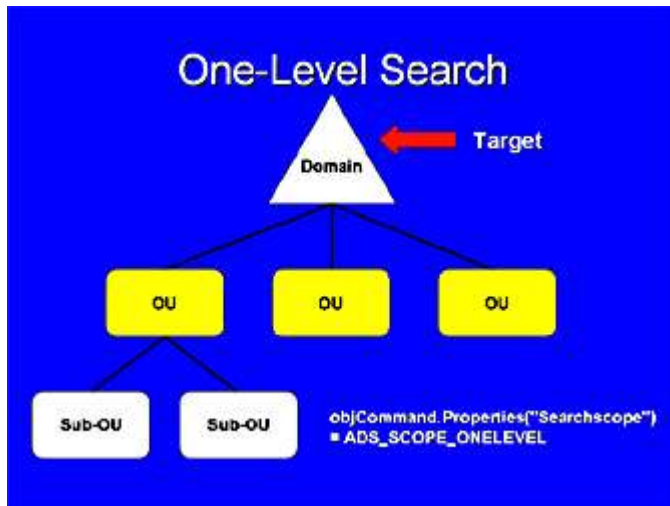
Later in your script, set the Searchscope property to the value of this constant:

```
objCommand.Properties("Searchscope") = ADS_SCOPE_BASE
```

One-Level Search

This is kind of an oddball search, but you might find it useful from time to time. Suppose you have an OU named Finance that has two child OUs: Accounting and Marketing. Suppose you perform a one-level search starting in the Finance OU. In that case, only the Accounting and Marketing OUs will be searched; the parent OU will *not* be searched. In addition, suppose the Accounting OU has two child OUs of its own: Receivables and Payables. Those sub-sub-OUs will not be searched, either; hence the term *one-level* search.

Here's another graphic that shows the containers that will be looked at when doing a one-level search beginning in the domain root. As you can see, the domain root is not searched, nor are any containers below the top-level OUs.



For a full-size version of this graphic, click [here](#).

To specify a one-level search, do two things. First, create a constant named `ADS_SCOPE_ONELEVEL` and set the value to 1:

```
Const ADS_SCOPE_ONELEVEL = 1
```

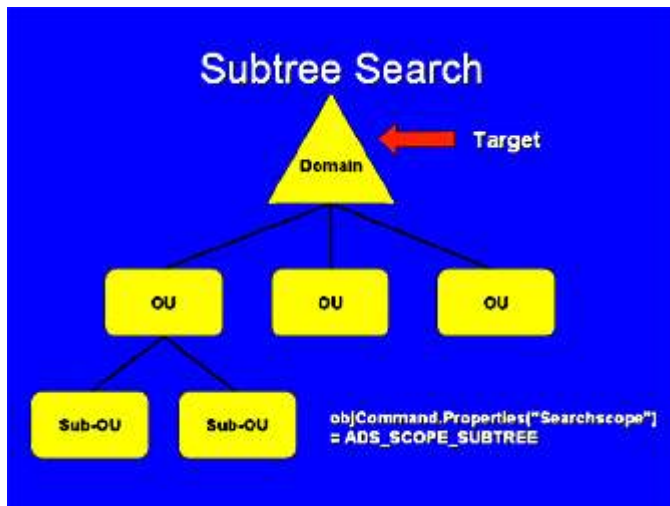
Later in your script, set the `Searchscope` property to the value of this constant:

```
objCommand.Properties("Searchscope") = ADS_SCOPE_ONELEVEL
```

Subtree Search

The subtree search is by far the most commonly-used search scope; in fact, this is the default scope that will be used if you don't otherwise specify a search scope. In a subtree search the entire subtree is searched: that includes the base container (for example, the Finance OU), and sub-containers (such as the Accounting and Marketing OUs), and any containers contained within those sub-containers. What does that mean? Well, for one thing, if you want to search your entire Active Directory all you need to do is specify a subtree search beginning in the Active Directory root; a search like that will search the root, then proceed to search any and all containers within the root. And because all containers are ultimately stored inside the root that means you'll end up searching all of Active Directory.

Or, to put it a bit more graphically, here's a diagram that shows the containers that will be searched when doing a subtree search beginning in the domain root (turns out that just happens to be *all* the containers in the domain).



For a full-size version of this graphic, click [here](#).

To specify a subtree search, do two things. First, create a constant named ADS_SCOPE_SUBTREE and set the value to 2:

```
Const ADS_SCOPE_SUBTREE = 2
```

Later in your script, set the Searchscope property to the value of this constant:

```
objCommand.Properties("Searchscope") = ADS_SCOPE_SUBTREE
```

[↑ Top of page](#)

Sort On

The Sort On property lets you specify how you want the returned data to be sorted. Want to sort the data by the last name (SN) of the user? Then use this code:

```
objCommand.Properties("Sort On") = "SN"
```

Prefer to sort the records by the department the user belongs to? Then use this code:

```
objCommand.Properties("Sort On") = "Department"
```

Note that you can sort on only a single value: you can't sort by department and then by last name within that department.

[↑ Top of page](#)

Creating a Query

There are actually two different ways to write Active Directory queries: you can use the LDAP syntax or the SQL syntax. In general, the two are equivalent: almost all searches that can be carried out using the LDAP syntax can be carried out using the SQL syntax, and vice-versa. We're going to focus on the SQL syntax because we find it both easier to use and much more familiar to the typical system administrator. (If you'd like to know

more about the LDAP syntax, take a look at the [Scripting Guys webcast](#) on searching Active Directory.)

For the rest of us, let's revisit the query we used to locate all the user accounts in a domain:

```
objCommand.CommandText = _
"SELECT Name FROM 'LDAP://dc=fabrikam,dc=com' WHERE objectCategory='user'"
```

Note. We should mention right away that Active Directory searches are read-only; because of that you're pretty much limited to using a SELECT query. Those of you familiar with database scripting might be wondering, "Is there any way to use a DELETE query to delete records or an UPDATE query to modify records?" There's a simple answer to that: no. But next month we'll show you a way that you can combine an Active Directory search with some standard ADSI scripting to do things like change the Department attribute for all members of the Accounting department to the Finance department.

For the most part, this looks like a typical SQL query. We're selecting something (**Name**) from somewhere (**'LDAP://dc=fabrikam,dc=com'**) based on specified criteria (**WHERE objectCategory = 'user'**). Pretty straightforward.

In fact, there are only a few things you have to know in order to write SQL queries to search Active Directory (assuming you're comfortable with writing SQL queries). To begin with, you must specify the names of all the attributes you want returned by this query. If you've done a lot of SQL or written a lot of WMI queries, you're no doubt used to using a query like this one as a shortcut method to return all the properties of an object:

```
SELECT * FROM ...
```

This won't work when querying Active Directory; SELECT * FROM is a valid query, but instead of returning all the attributes of an object this query returns only the ADsPath attribute. What if you want to return the user CN (common name), department, and job title? In that case, you'll have to include all three of those attributes in your query, like so:

```
objCommand.CommandText = _
"SELECT CN, Department, Title FROM 'LDAP://dc=fabrikam,dc=com' " & _
"WHERE objectCategory='user'"
```

Note. What if you want to return a couple hundred attributes for a user account? Although you probably *could* write a SQL query that would return all that information we don't recommend it. Instead, you're better off returning just the ADsPath, using the ADsPath to connect to each object, and then returning all the information for each object individually. That's something we'll address in Part 2 of this series.

Second, you must use the ADsPath to specify the starting location for a search. Typically you'll want to start your search in the root of the domain and then go from there. To do that, you simply specify the ADsPath to the domain root, *making sure to enclose the path in single quote marks*:

```
'LDAP://dc=fabrikam,dc=com'
```

But what if you don't want to start the search from the domain root, what if you want to start the search in a specific OU? That's no problem; just use the ADsPath to that OU:

```
'LDAP://ou=Finance,dc=fabrikam,dc=com'
```

What if you'd like to search the entire *forest*? In that case, just specify the ADsPath to the global catalog:

```
'GC://dc=fabrikam,dc=com'
```

Note. Provider names are case-sensitive. Don't type in ldap or gc; you'll be very disappointed.

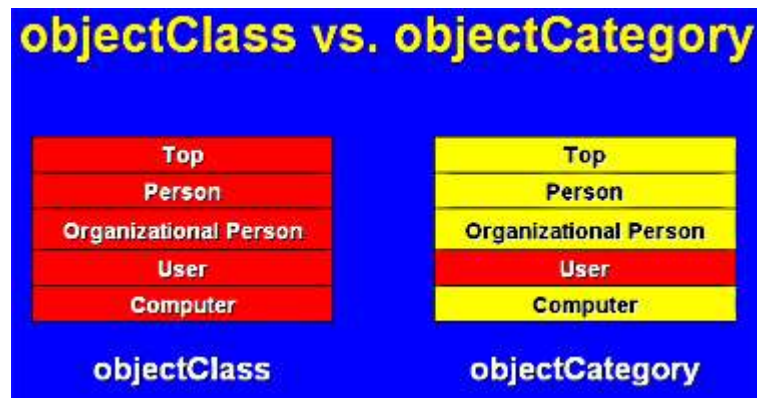
Last, but surely not least, is the WHERE clause. Here you specify exactly what it is you're searching for. One thing that will (or at least *should*) be included in nearly every WHERE clause you ever write is the objectCategory. The objectCategory indicates the type of object we're searching for. For example, in this query we're looking solely for user objects:

```
objCommand.CommandText = _
  "SELECT CN, Department, Title FROM 'LDAP://dc=fabrikam,dc=com' " & _
  "WHERE objectCategory='user'"
```

In this one, we're after *either* user objects or computer objects:

```
objCommand.CommandText = _
  "SELECT CN, Department, Title FROM 'LDAP://dc=fabrikam,dc=com' " & _
  "WHERE objectCategory='user' OR objectCategory='computer'"
```

Note that you will often see search scripts that specify an objectClass rather than an objectCategory. The difference between objectClass and objectCategory has to do with Active Directory inheritance. We won't discuss inheritance in any detail today; we'll just mention that most Active Directory classes are derived from other classes. For example, the following illustration (which has been simplified a bit for our purposes) shows a class inheritance tree in Active Directory. At the top of the tree is the aptly-named Top class. Right below it is the Person class. The Person class contains all the properties of the Top class *plus* a few properties not found in Top. Right below the Person class is the Organizational Person class; as you might expect, Organizational Person contains all the attributes found in the Top class, all the attributes found in the Person class, *plus* a few unique attributes all its own.



Why does that matter to us? Well, when you search for an objectClass, you get back all the objects found in the inheritance tree; thus searching for **objectClass='user'** will return users and organization persons and everything else that inherits from computer. By contrast, searching for **objectCategory='user'** returns *only* computer objects. *That's* why this matters to us.

[↕ Top of page](#)

Executing the Query

As soon as your query is finished all you have to do is call the Execute method to start the data retrieval process:

```
Set objRecordSet = objCommand.Execute
```

As you can see, as soon as you get all the properties and parameters specified the search itself is easy.

[↕ Top of page](#)

Looping Through the Returned Recordset

After you execute the query, you'll get your data back in the form of a recordset. To do something with that data, you simply need to cycle through all the records in that recordset. To do *that*, use the MoveFirst method to make sure you're starting with the first record. (To tell you the truth, we're not sure you *have* to do this, but we'd recommend it, just to be on the safe side.) Perform a Do Until loop until you reach the end of the recordset (EOF), and do something with each record as you loop through. For example, this code simply echoes the value of the Name attribute for each item in the recordset:

```
objRecordSet.MoveFirst
Do Until objRecordSet.EOF
    wscript.Echo objRecordSet.Fields("Name").Value
    objRecordSet.MoveNext
Loop
```

Two things to keep in mind here. First, what you're really doing is echoing the value of the Name field. What if you wanted to echo the value of the Title field (assuming you included Title in your query)? Well:

```
wscript.Echo objRecordSet.Fields("Title").Value
```

Second, note this crucial line of code:

```
objRecordSet.MoveNext
```

This line of code tells the script to move to the next record in the recordset. If you forget to include this code (um, not that any of the Scripting Guys have ever done this) your script will simply echo the value of the first record in the recordset for ever and ever. That's because you have to explicitly instruct the script to move to the next record.

[↕ Top of page](#)

Stay Tuned for Our Next Exciting Episode

That should be enough to get you started with writing scripts that search Active Directory. With the basics out of the way, next month we'll explore in more depth some of the things you can actually search for. Until then, you might want to look at the [Scripting Guys webcast](#) on searching Active Directory, and maybe check out some of the sample scripts found in the [Script Center Script Repository](#). As for the Scripting Guys, we've decided to take a spur-of-the-moment trip to the jungles of South America. Why? Oh, no reason, really

[↕ Top of page](#)

[Manage Your Profile](#)

© 2008 Microsoft Corporation. All rights reserved. [Terms of Use](#) | [Trademarks](#) | [Privacy Statement](#) | [Contact Us](#)

