
Appendix E: Using ADSI Programming to Automate Administrative Tasks



Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, places or events is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2000 Microsoft Corporation. All rights reserved.

Microsoft, Active Directory, BackOffice, FrontPage, IntelliMirror, PowerPoint, Visual Basic, Visual Studio, Win32, Windows, Windows Media, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the U.S.A. and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Objectives

After completing this lab, you will be able to:

- Create multiple Active Directory™ directory service objects.
- Modify multiple Active Directory objects.
- Delete multiple Active Directory objects.

Prerequisites

Before working on this lab, you must have:

- A basic understanding of the Active Directory directory service.
- A basic understanding of the Lightweight Directory Access Protocol (LDAP).
- Experience writing batch files.

Lab Setup

To complete this lab, you need the following:

- A computer running Microsoft® Windows® 2000 Server configured as a domain controller.
- Cr_user.vbs, Mod_user.vbs, Del_user.vbs, and Users.txt found in the appendix folder on the Student Materials compact disc.

For More Information

For more information about using Active Directory Service Interfaces (ADSI), see <http://msdn.microsoft.com/developer/windows2000/adsi>.

Lab Design

The lab is designed so that the student uses three ADSI scripts to create, modify, and delete multiple user objects. The first script retrieves data from a text file, which is modified by the student to use with the other two scripts. Each exercise builds on the work from the previous exercise.

Scenario

You are an enterprise administrator who must add, modify, and delete hundreds of users, groups, and organizational units (OUs) from the network every day. You want to automate most of these tasks by using the Windows Script Host and ADSI scripting.

Your objective is to:

- Create a Human Resources OU.
- Add several users and their job titles in the Human Resources OU.
- Modify several users' job titles.
- Delete several users.

Estimated time to complete this lab: 60 minutes

Background

ADSI is a set of generic interfaces that allow you to access and manipulate different directory services. This lab will focus on the Windows 2000 Active Directory directory service.

There are several different ways to access Active Directory. It is recommended that you use ADSI as the strategic application programming interface (API) to access Active Directory. ADSI uses the LDAP protocol to communicate with Active Directory.

When using a script to gain access to an object in Active Directory, you must provide the LDAP namespace, the programmatic identifier, which is ProgID, and the path to the object, which is ADsPath. Then you will bind to the domain to provide a reference point for creating and modifying Active Directory objects.

The Namespace

A *namespace* is simply a set of names in which all names are unique. For example, files stored on a disk drive reside in the *file system namespace*. The unique name of a file is based on where it is stored in the file system namespace, for example:

```
C:\public\documents\adsi\adsi_spec_v3.doc
```

Directory service namespaces also identify the objects they contain by unique names, which are usually based on the location in the directory where the object can be found. For example, in Active Directory, a given object might have a name like this:

```
CN=jsmith, OU=Sales, DC=nwtraders, DC=msft
```

The ProgID and ADsPath

ADSI defines a naming convention that can uniquely identify an ADSI object in a heterogeneous environment. These names are called *ADsPath strings*. ADsPath strings consist of a ProgID (for example LDAP:, WinNT:) and the provider's specific path.

The following table provides examples of ADsPath strings.

ADSI object	Path
Organizational unit in the nwtraders domain	LDAP://OU=Sales, DC=nwtraders, DC=msft
Exchange object on Exchange Server	LDAP://exch01/O=Microsoft
jsmith user in the Sales OU of the nwtraders domain	LDAP://CN=jsmith, OU=Sales, DC=nwtraders, DC=msft
comp1 in the Redmond domain	WinNT://REDMOND/comp1, computer
alice, a local user on the comp1 computer	WinNT://REDMOND/comp1/alice

Notice that each ProgID is synonymous with a unique namespace.

The Root DS Entry

The LDAP standard, which is defined in RFC 2251, requires that all LDAP directories maintain a special entry, called the *rootDSE object*. This entry provides a set of standard operational attributes that the user can read to find out fundamental characteristics of the directory and the server. The rootDSE can also provide any number of attributes specific to a vendor.

One of the standard operational attributes is *defaultNamingContext*. This attribute contains the distinguished name of the directory root. In Windows 2000, this is the distinguished name of the domain container at the root of the current tree. By reading the defaultNamingContext attribute from the rootDSE, you can determine the domain to which you are logged on at run time.

ADSI provides a special mechanism for binding to the rootDSE by using the ADSpath "LDAP:// rootDSE."

```
set Root = GetObject("LDAP://RootDSE")
```

Next, you retrieve the default naming context:

```
DomainPath = Root.Get("DefaultNamingContext")
```

Then you bind to the root of the domain:

```
Set Domain = GetObject("LDAP://" & DomainPath)
```

These three lines of code appear first in most of the ADSI scripts that you create. This code sets the reference point in your script so that you can create users, groups, and organizational units in the domain.

You should never use a specific server name. Additionally, under most circumstances, you should avoid binding to a single server. Active Directory supports serverless binding (as shown in the previous example), which means it is not necessary to specify a server if you are accessing Active Directory in the domain of the user who is logged on. When processing your bind call, ADSI finds the Windows 2000 domain controller in the domain of the user who is currently logged on.

Creating Organizational Units

You can now use the domain object to create other objects such as organizational units. You need to create the Human Resources OU. The sample code to create the OU object is as follows:

```
Set ou_HR = Domain.Create("organizationalUnit", "OU=Human  
Resources")  
ou_HR.Put "Description", "Human Resources"  
ou_HR.SetInfo
```

The **Create** method accepts the class name and the name of the new object. At this point, the object is not committed to Active Directory. You will, however, have an ADSI/COM object reference on the client. This ADSI object enables you to set or modify attributes by using the **Put** method. The **Put** method accepts the attribute name and the value of the attribute. Still, no objects are committed to the directory; everything is cached on the client side.

When you call the **SetInfo** method, the changes (in this case, object creation and attribute modification) are committed to Active Directory. These changes are *transacted*, which means that you see either the new object with all of the attributes you have set, or no object at all. Open Active Directory Users and Computers to verify that the organizational unit objects appear.

Creating User Accounts

You now need to create users in the organizational unit. The sample code for creating a user is as follows:

```
Set usr = ou_HR.Create("user", "CN=James Smith")
usr.Put "samAccountName", "jasmith"
usr.Put "userPrincipalName", "jasmith" & "@" &
Domain.Get("name")
usr.AccountDisabled = False
usr.SetInfo
usr.SetPassword "password"
usr.SetInfo
```

Notice that you must specify the `samAccountName`. The `samAccountName` is a mandatory attribute for the user class. Before you can create an instance of an object, you must set all mandatory attributes. The user account's `samAccountName` is used to log on from computers running Microsoft Windows 95, Microsoft Windows 98, or Microsoft Windows NT®. Computers running Windows 2000 understand the `samAccountName`. However, when both the client and domain controller are running Windows 2000, you can log on by using the user principal name. In this example, James's user principal name is set to `jasmith@your_domain`. If James moves to a different domain in the forest, he can continue to use his user principal name.

You enable the user's account by setting the `AccountDisabled` property to `FALSE`. Administrators are also able to assign a password by using the **SetPassword** method. However, the **SetPassword** method works only if the object has been created in the directory. You need to call the **SetInfo** method before attempting to set the user's password.

Note The sample code to create a new group is as follows:

```
Set Users = GetObject("LDAP://CN=USERS," & DomainPath)
Set grp = Users.Create("group", "CN=HR Admins")
grp.Put "samAccountName", "HRadmins"
grp.SetInfo
```

In the first line, the ADSI object is set to the Users container. The group, HR Admins, is then created in that container. The `samAccountName` is a mandatory attribute, which is added to provide backward compatibility. In this example, tools in Microsoft Windows NT version 4.0, such as User Manager, recognize HRadmins instead of HR Admins.

In this lab, rather than creating each object individually as shown in the previous examples of the sample code, the script retrieves the user names from a text file and creates several objects by using a loop construct.

Exercise 1 Creating Objects by Using ADSI

Goal

In this exercise, you will use a script to create the Human Resources OU and several user accounts in the OU. The user names and job titles will be copied into the script from a text file.

Tasks	Detailed Steps
1. Open the Users.txt file and create five additional users, following the convention illustrated by the three existing users.	a. Open Users.txt in the appendix folder on the Student Materials compact disc. b. Add five more users to the end of the text file by using the following format: <code>username:Project Lead:Full Name</code> c. Save the file, and then close Notepad.
2. Examine, test and verify the Cr_user.vbs script.	a. Right-click Cr_user.vbs in the appendix folder, and then click Edit. b. Examine the annotated code, particularly those sections that can be customized to meet your needs. c. Close Notepad. d. Double-click Cr_user.vbs, and then click OK in any Windows Script Host message boxes that appear. e. Open Active Directory Users and Computers, and then verify that all of the user accounts have been created in the Human Resources OU.

Exercise 2 Modifying Objects by Using ADSI

Goal

In this exercise, you will use a script to edit several user accounts in the OU. The user names and job titles will be copied into the script from a text file.

Tasks	Detailed Steps
1. Open the Users.txt file and change several of the job titles, and then save the file as Change.txt.	a. Open Users.txt in the appendix folder on the Student Materials compact disc. b. Change the Project Lead job titles to Project Manager, by using the following format: <code>username:Project Manager</code> c. Save the file as Change.txt, and then close Notepad.
2. Examine, test and verify the Mod_user.vbs script.	a. Right-click Mod_user.vbs in the appendix folder, and then click Edit. b. Examine the annotated code, particularly those sections that can be customized to meet your needs. c. Close Notepad. d. Double-click Mod_user.vbs, and then click OK in any Windows Script Host message boxes that appear. e. Open Active Directory Users and Computers, and then verify that the user accounts have been modified in the Human Resources OU.

Exercise 3 Deleting Objects by Using ADSI

Goal

In this exercise, you will use a script to delete several user accounts in the OU. The user names will be copied into the script from a text file.

Tasks	Detailed Steps
1. Open the Change.txt file, remove those user names that you do not want to delete, and then save the file as Remove.txt.	a. Open Change.txt in the appendix folder on the Student Materials compact disc. b. Change the file to include only those user names that you want to delete, by using the following format: username c. Save the file as remove.txt, and then close Notepad.
2. Examine, test and verify the Del_user.vbs script.	a. Right-click Del_user.vbs in the appendix folder, and then click Edit. b. Examine the annotated code, particularly those sections that can be customized to meet your needs. c. Close Notepad. d. Double-click Del_user.vbs, and then click OK in any Windows Script Host message boxes that appear. e. Open Active Directory Users and Computers, and then verify that the user accounts have been modified in the Human Resources OU.