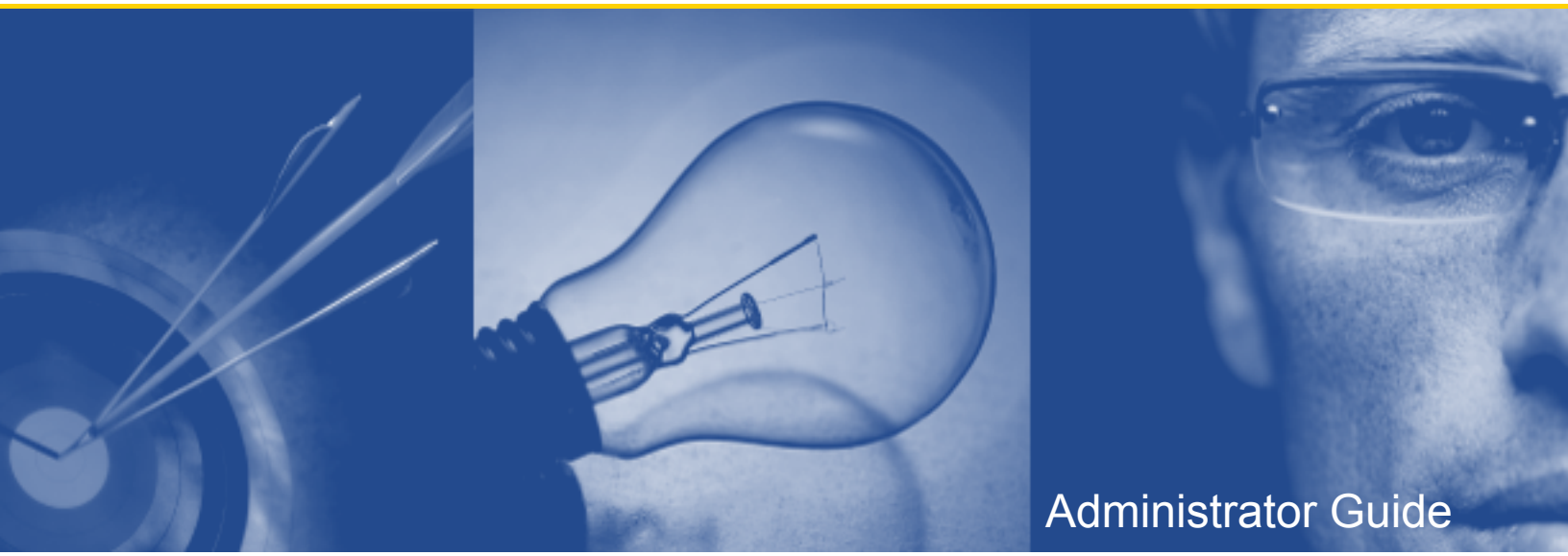
ActiveRoles Management Shell for Active Directory

Version 1.2



© 2009 Quest Software, Inc.
ALL RIGHTS RESERVED.

This guide contains proprietary information protected by copyright. The software described in this guide is furnished under a software license or nondisclosure agreement. This software may be used or copied only in accordance with the terms of the applicable agreement. No part of this guide may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's personal use without the written permission of Quest Software, Inc.

If you have any questions regarding your potential use of this material, contact:

Quest Software World Headquarters
LEGAL Dept
5 Polaris Way
Aliso Viejo, CA 92656
USA
www.quest.com
email: legal@quest.com

Refer to our Web site for regional and international office information.

Trademarks

Quest, Quest Software, the Quest Software logo, Aelita, Akonix, AppAssure, Benchmark Factory, Big Brother, ChangeAuditor, DataFactory, DeployDirector, ERDisk, Foglight, Funnel Web, GPOAdmin, I/Watch, Imceda, InLook, IntelliProfile, InTrust, Invertus, IT Dad, I/Watch, JClass, Jint, JProbe, LeccoTech, LiteSpeed, LiveReorg, MessageStats, NBSpool, NetBase, Npulse, NetPro, PassGo, PerformaSure, Quest Central, SharePlex, Sitraka, SmartAlarm, Spotlight, SQL LiteSpeed, SQL Navigator, SQL Watch, SQLab, Stat, StealthCollect, Tag and Follow, Toad, T.O.A.D., Toad World, vAnalyzer, vAutomator, vControl, vConverter, vEssentials, vFoglight, vMigrator, vOptimizer Pro, vPackager, vRanger, vRanger Pro, vReplicator, vSpotlight, vToad, Vintela, Virtual DBA, VizionCore, Vizioncore vAutomation Suite, Vizioncore vEssentials, Xaffire, and XRT are trademarks and registered trademarks of Quest Software, Inc. in the United States of America and other countries. Other trademarks and registered trademarks used in this guide are property of their respective owners.

Disclaimer

The information in this document is provided in connection with Quest products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Quest products. **EXCEPT AS SET FORTH IN QUEST'S TERMS AND CONDITIONS AS SPECIFIED IN THE LICENSE AGREEMENT FOR THIS PRODUCT, QUEST ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL QUEST BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF QUEST HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Quest makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Quest does not make any commitment to update the information contained in this document.

Quest ActiveRoles Management Shell for Active Directory - Administrator Guide
Updated - May 13, 2009
Software Version - 1.2

CONTENTS

INTENDED AUDIENCE 5
CONVENTIONS 5
ABOUT QUEST SOFTWARE, INC. 6
CONTACTING QUEST SOFTWARE 6
CONTACTING QUEST SUPPORT 6
INTRODUCTION 7
USING ACTIVEROLES MANAGEMENT SHELL 8
INSTALLING AND OPENING ACTIVEROLES MANAGEMENT SHELL 8
GETTING HELP 10
CMDLET NAMING CONVENTIONS 11
PARAMETERS 11
SYNTAX 14
PIPELINING 15
ALIASES 16
WHAT'S NEW IN VERSION 1.2 18
TROUBLESHOOTING 20
CMDLET REFERENCE - ACTIVE DIRECTORY 21
CONNECT-QADSERVICE 22
DISCONNECT-QADSERVICE 25
GET-QADUSER 26
SET-QADUSER 42
NEW-QADUSER 49
DISABLE-QADUSER 53
ENABLE-QADUSER 55
UNLOCK-QADUSER 57
DEPROVISION-QADUSER 59
GET-QADGROUP 61
SET-QADGROUP 71
NEW-QADGROUP 75
GET-QADGROUPMEMBER 79
ADD-QADGROUPMEMBER 85
REMOVE-QADGROUPMEMBER 87
GET-QADMEMBEROF 89
GET-QADCOMPUTER 95
GET-QADOBJECT 105
SET-QADOBJECT 114
NEW-QADOBJECT 117
MOVE-QADOBJECT 120

Rename-QADObject 122
Remove-QADObject 124
Restore-QADDeletedObject 126
New-QADPasswordSettingsObject 128
Get-QADPasswordSettingsObject 132
Get-QADPasswordSettingsObjectAppliesTo 143
Add-QADPasswordSettingsObjectAppliesTo 149
Remove-QADPasswordSettingsObjectAppliesTo 151
Get-QADPermission 153
Add-QADPermission 158
Remove-QADPermission 163
Get-QADObjectSecurity 165
Set-QADObjectSecurity 167
Get-QADRootDSE 170
Cmdlet Reference - ActiveRoles Server 172
Get-QARSAccessTemplate 173
Get-QARSAccessTemplateLink 180
Set-QARSAccessTemplateLink 190
New-QARSAccessTemplateLink 193
Remove-QARSAccessTemplateLink 196
Get-QARSOperation 198
Get-QARSApprovalTask 207
Approve-QARSApprovalTask 213
Reject-QARSApprovalTask 215
Cmdlet Reference - Utility 217
Convert-QADAttributeValue 218
Get-QADPSSnapinSettings 220
Set-QADPSSnapinSettings 223

Intended Audience

This document has been prepared to assist you in becoming familiar with the Quest ActiveRoles Management Shell for Active Directory. The Administrator Guide contains the information required to install and use the Quest ActiveRoles Management Shell for Active Directory. It is intended for network administrators, consultants, analysts, and any other IT professionals using the product.

Conventions

In order to help you get the most out of this guide, we have used specific formatting conventions. These conventions apply to procedures, icons, keystrokes and cross-references.

ELEMENT	CONVENTION
Select	This word refers to actions such as choosing or highlighting various interface elements, such as files and radio buttons.
Bolded text	Interface elements that appear in Quest Software products, such as menus and commands.
Italic text	Used for comments.
Bold Italic text	Used for emphasis.
Blue text	Indicates a cross-reference. When viewed in Adobe® Reader®, this format can be used as a hyperlink.
	Used to highlight additional information pertinent to the process being described.
	Used to provide Best Practice information. A best practice details the recommended course of action for the best result.
	Used to highlight processes that should be performed with care.
+	A plus sign between two keystrokes means that you must press them at the same time.
|	A pipe sign between elements means that you must select the elements in that particular sequence.

About Quest Software, Inc.

Quest Software, Inc., a two-time winner of Microsoft's Global Independent Software Vendor Partner of the Year award, delivers innovative products that help organizations get more performance and productivity from their applications, databases Windows infrastructure and virtual environments. Through a deep expertise in IT operations and a continued focus on what works best, Quest helps more than 100,000 customers worldwide meet higher expectations for enterprise IT. Quest's Windows management solutions simplify, automate secure and extend Active Directory, Exchange Server, SharePoint, SQL Server, .NET and Windows Server as well as integrating Unix, Linux and Java into the managed environment. Quest Software can be found in offices around the globe and at www.quest.com.

Contacting Quest Software

Email	info@quest.com
Mail	Quest Software, Inc. World Headquarters 5 Polaris Way Aliso Viejo, CA 92656 USA
Web site	www.quest.com

Refer to our Web site for regional and international office information.

Contacting Quest Support

Quest Support is available to customers who have a trial version of a Quest product or who have purchased a commercial version and have a valid maintenance contract. Quest Support provides around the clock coverage with SupportLink, our web self-service. Visit SupportLink at http://support.quest.com/

From SupportLink, you can do the following:

- Quickly find thousands of solutions (Knowledgebase articles/documents).
- Download patches and upgrades.
- Seek help from a Support engineer.
- Log and update your case, and check its status.

View the ***Global Support Guide*** for a detailed explanation of support programs, online services, contact information, and policy and procedures. The guide is available at http://support.quest.com/pdfs/Global Support Guide.pdf.

Note: This document is only available in English.

Introduction

Quest ActiveRoles Management Shell for Active Directory is an Active Directory-specific automation and scripting shell that provides a command-line management interface for administering directory data either via Quest ActiveRoles Server or by directly accessing Active Directory domain controllers. ActiveRoles Management Shell is built on Microsoft Windows PowerShell technology.

This document is designed to introduce new users to ActiveRoles Management Shell. The document provides information on the basic concepts and features of ActiveRoles Management Shell, and includes reference topics about the commands (cmdlets) that can be run in ActiveRoles Management Shell. The document examines:

- Installing and using ActiveRoles Management Shell
- ActiveRoles Management Shell command-line tools

ActiveRoles Management Shell is implemented as a Windows PowerShell snap-in, providing an extension to the Windows PowerShell environment. To get acquainted with the basic features of Windows PowerShell, refer to the Windows PowerShell Getting Started Guide, which you can access at http://msdn.microsoft.com/en-us/library/aa973757.aspx. For more detailed information on Windows PowerShell, see the Windows PowerShell Primer document, which is included with the Windows PowerShell installation.

As the commands provided by ActiveRoles Management Shell conform to the Windows PowerShell standards, and are fully compatible with the default command-line tools that come with Windows PowerShell, the information found in the above documents of Microsoft's is fully applicable to ActiveRoles Management Shell.

Using ActiveRoles Management Shell

Quest ActiveRoles Management Shell for Active Directory, built on Microsoft Windows PowerShell technology, provides a command-line interface that enables automation of directory data-related administrative tasks. With ActiveRoles Management Shell, administrators can manage directory objects such as users and groups. Thus, they can create new users and groups, modify user properties, and add or remove members from groups.

The management operations are performed either via the Quest ActiveRoles Server proxy service or by directly accessing directory data on domain controllers. In both cases, ActiveRoles Management Shell provides a flexible scripting platform that can reduce the complexity of current Microsoft Visual Basic scripts. Tasks that previously required many lines in Visual Basic scripts can now be done by using as little as one line of code in ActiveRoles Management Shell.

By accessing the directory services through the Quest ActiveRoles Server proxy service, ActiveRoles Management Shell makes it possible to take full advantage of the security, workflow integration and reporting benefits of ActiveRoles Server. In this way, the directory data modifications made by ActiveRoles Management Shell are supplemented and restricted by the data validation, provisioning and deprovisioning rules enforced by ActiveRoles Server.

The ActiveRoles Management Shell command-line tools (cmdlets), like all the Windows PowerShell cmdlets, are designed to deal with objects—structured information that is more than just a string of characters appearing on the screen. The cmdlets do not use text as the basis for interaction with the system, but use an object model that is based on the Microsoft .NET platform. In contrast to traditional, text-based commands, the cmdlets do not require the use of text-processing tools to extract specific information. Rather, you can access portions of the data directly by using standard Windows PowerShell object manipulation commands.

Installing and Opening ActiveRoles Management Shell

Installation Requirements

Before you install ActiveRoles Management Shell, ensure that your system has the following software installed:

- Windows XP Service Pack 2, Windows 2003 Service Pack 1, or later versions of Windows
- Microsoft .NET Framework 3.5 Service Pack 1, or a later version of .NET Framework
- Microsoft Windows PowerShell 1.0

If you are planning to manage Terminal Services user properties by using cmdlets on a Windows XP or Windows Vista based computer, then you have to install additional administration tools on that computer:

- On a computer running a 32-bit edition of Windows XP, install Windows Server 2003 Service Pack 2 Administration Tools Pack for x86 editions, available for download at http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=86b71a4f-4122-44af-be79-3f101e533d95
- On a computer running a 64-bit edition of Windows XP, install Windows Server 2003 Service Pack 2 Administration Tools Pack for x64 editions, available for download at http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=514bd06e-f3bc-4054-8429-c49f51e2190b

- On a computer running a 32-bit edition of Windows Vista, install Microsoft Remote Server Administration Tools for Windows Vista, available for download at http://www.microsoft.com/downloads/details.aspx?FamilyID=9ff6e897-23ce-4a36-b7fc-d52065de9960&DisplayLang=en
- On a computer running a 64-bit edition of Windows Vista, install Microsoft Remote Server Administration Tools for Windows Vista for x64-based Systems, available for download at http://www.microsoft.com/downloads/details.aspx?FamilyID=d647a60b-63fd-4ac5-9243-bd3c497d2bc5&DisplayLang=en

In addition, on a Windows Vista based computer, you need to configure the administration tools to enable ADSI Terminal Services extensions. For instructions, see http://blogs.technet.com/askds/archive/2008/03/31/rsat-and-aduc-getting-the-terminal-services-tabs-to-appear-in-ad-users-and-computers.aspx

Installing Microsoft .NET Framework

For information on how to download and install Microsoft .NET Framework, see .NET Framework Developer Center at http://msdn.microsoft.com/en-us/netframework/default.aspx

Installing Microsoft Windows PowerShell

For information on how to download and install Microsoft Windows PowerShell, see Microsoft Knowledge Base article 926139, Windows PowerShell 1.0 English Language Installation Packages for Windows Server 2003 and for Windows XP, http://support.microsoft.com/?kbid=926139

If you are running Windows Server 2008, to install Windows PowerShell, perform the following steps:

1. Click **Start**, and then click **Control Panel**.
2. In Control Panel, double-click **Administrative Tools**.
3. In **Administrative Tools**, double-click **Server Manager**.
4. In **Server Manager**, in the console tree, click **Features**, and then in the details pane, click **Add Features**.
5. In the Add Features Wizard, select **Windows PowerShell**, and then complete the wizard.

Installing ActiveRoles Management Shell

To install ActiveRoles Management Shell

1. Run the **Setup.exe** file, included with the ActiveRoles Management Shell distribution package.
2. Follow the instructions on the installation wizard pages.

Opening ActiveRoles Management Shell

You can open ActiveRoles Management Shell by using either of the following procedures. Each procedure loads the ActiveRoles Management Shell snap-in into Windows PowerShell. If you do not load the ActiveRoles Management Shell snap-in before you run a command (cmdlet) provided by that snap-in, you will receive an error.

To open ActiveRoles Management Shell from the Programs menu

- Select **Start | All Programs | Quest Software | ActiveRoles Management Shell for Active Directory**.

To add the ActiveRoles Management Shell snap-in from Windows PowerShell

1. Select **Start | All Programs | Windows PowerShell 1.0 | Windows PowerShell**.
2. At the Windows PowerShell prompt, enter the following command:

```
Add-PSSnapin Quest.ActiveRoles.ADManagement
```

Upon the shell start, the console display a message stating that a certain file published by Quest Software is not trusted on your system. This security message indicates that the certificate the file is digitally signed with is not trusted on your computer, so the console requires you to enable trust for the certificate issuer before the file can be run. Press either **R** (Run once) or **A** (Always run). To prevent this message from appearing in the future, it is advisable to choose the second option (**A**).

Getting Help

ActiveRoles Management Shell uses the Windows PowerShell help cmdlets to assist you in finding the appropriate information to accomplish your task. The following table provides some examples of how to use the **Get-Help** and **Get-Command** cmdlets to access the help information that is available for each cmdlet in ActiveRoles Management Shell.

COMMAND	DESCRIPTION
Get-Help	When you use **Get-Help** without any parameters, you are presented with basic instructions on how to use the help system in Windows PowerShell, including Help for ActiveRoles Management Shell.
Get-Help <Cmdlet>	When you use **Get-Help** with the name of a cmdlet as an argument, you are presented with the help information for that cmdlet. For example, to retrieve the help information for the **Connect-QADService** cmdlet, use the following command: • Get-Help Connect-QADService
Get-Command	**Get-Command** without any parameters lists all the cmdlets that are available to the shell. You can use the **Get-Command** cmdlet with the **Format-List** or **Format-Table** cmdlet to provide a more readable display. For example, use **Get-Command | Format-List** to display the output in a list format.
Get-Command <Cmdlet>	When you use **Get-Command** with the name of a cmdlet as an argument, you are presented with information about the parameters and other components of that cmdlet. The <*Cmdlet*> entry allows for wildcard character expansion. For example, to retrieve information about the cmdlets with the names ending in **Member**, you can use the following command: • Get-Command *Member
Get-Command -Noun <CmdletNoun>	**Get-Command -Noun** <*CmdletNoun*> lists all the cmdlets with the names that include the specified noun. <*CmdletNoun*> allows for wildcard character expansion. Thus, you can use the following command to list all the cmdlets provided by ActiveRoles Management Shell: • Get-Command -Noun QA*

Cmdlet Naming Conventions

All cmdlets are presented in verb-noun pairs. The verb-noun pair is separated by a hyphen (-) without spaces, and the cmdlet nouns are always singular. The verb refers to the action that the cmdlet performs. The noun identifies the entity on which the action is performed. For example, in the **Set-QADUser** cmdlet name, the verb is **Set** and the noun is **QADUser**. All ActiveRoles Management Shell cmdlets have the nouns prefixed with **QA**, to distinguish the ActiveRoles Management Shell cmdlets from those provided by Windows PowerShell itself or by other PowerShell snap-ins.

You can use the following command to list all cmdlets found in ActiveRoles Management Shell:

```
get-command Quest.ActiveRoles.ADManagement\*
```

Tab Expansion to Auto-complete Names

ActiveRoles Management Shell provides a way to complete command and parameter names automatically, thus speeding up command entry. You can fill in cmdlet names and parameters by pressing the TAB key.

To use tab expansion on a cmdlet name, type the entire first part of the name (the verb) and the hyphen that follows it, and then press TAB. The shell will complete the cmdlet name if a matching cmdlet is found. If multiple matching cmdlet names exist, repeatedly pressing TAB will cycle through all of the available choices. You can fill in more of the name for a partial match. The following example shows how you can use tab expansion when you enter a cmdlet name:

```
New-QAD <TAB>
```

As you press the TAB key in this example, the shell cycles through all the cmdlet names that begin with **New-QAD**.

You can also use tab expansion when you want the shell to complete the partial parameter name that you have entered. In this case, you must specify the full cmdlet name, either by typing it in directly or by using tab expansion. The following example shows how you can use tab expansion when you enter a parameter name:

```
Add-QADGroupMember -m <TAB>
```

As you press the TAB key in this example, the shell completes the *Member* parameter on the **Add-QADGroupMember** cmdlet.

Parameters

Cmdlets use parameters to take information necessary for completing their tasks. Parameters are string elements that follow the name of a cmdlet, either identifying an object and its attributes to act upon, or controlling how the cmdlet performs its task. The name of the parameter is preceded by a hyphen (-) and followed by the value of the parameter as follows:

```
Verb-Noun -ParameterName <ParameterValue>
```

In this example, the hyphen in front of the parameter name indicates that the word immediately following the hyphen is a parameter passed to the cmdlet and the next separate string after the parameter name is the value of the parameter.

Parameter Details

The information displayed by the **Get-Help** cmdlet includes the Parameters section (also called metadata) on each parameter. The following example is an excerpt from the output of the **Get-Help Connect-QADService -Full** command:

```
PARAMETERS
      -Proxy

            Required?                       false
            Position?                       named
            Default value
            Accept pipeline input?          false
            Accept wildcard characters?      false
```

This example from the **Connect-QADService** cmdlet includes some very specific details about the *Proxy* parameter. Some cmdlets may not include such details. However, most cmdlets do include some settings for each parameter as described in the following table.

SETTING	DESCRIPTION
Required?	Indicates whether the cmdlet will run if you do not supply the parameter. When *Required?* is set to True, the shell prompts you for the parameter if you do not supply a value for this parameter.
Position?	Indicates whether you must specify the parameter name in front of the parameter value. When *Position?* is set to Named, the parameter name is required. When *Position?* is set to an integer, the name is not required, only the value (see "Positional Parameters" later in this section).
Default value	Indicates the default value for this parameter if no other value is provided.
Accept pipeline input?	Indicates whether the parameter can receive its value as an input through a pipeline from another cmdlet (see "Pipelining" later in this document).
Accept wildcard characters?	Indicates whether the value of this parameter can contain wildcard characters and can be matched to multiple objects.

Positional Parameters

A positional parameter lets you specify the parameter's value without specifying the parameter's name. A positional parameter has the Position attribute set to an integer in the metadata. This integer indicates the position on the command line where the cmdlet can find the parameter's value.

An example of a positional parameter is the *Identity* parameter. This parameter is always in position 1 if it is available on a cmdlet. The following two commands perform the same task: resetting the password for the user identified by the logon name in the form domain\name:

```
Set-QADUser -Identity 'domain\jsmith' -UserPassword 'P@ssword'
Set-QADUser 'domain\jsmith' -UserPassword 'P@ssword'
```

If a parameter is not a positional parameter, it is considered to be a named parameter. When you enter a command on the command line, you must type the parameter name for a named parameter.

Switch Parameters

Switch parameters are used to set a state for the execution of a cmdlet. A switch parameter does not require a value. If you specify a switch parameter on a command line, the parameter evaluates to True. If you do not specify a switch parameter, it evaluates to False. For example, the *Proxy* parameter on the **Connect-QADService** cmdlet allows you to specify whether to access directory data via ActiveRoles Server (*-Proxy* is added on the command line) or by connecting directly to a domain controller (*-Proxy* is omitted).

Identity

The *Identity* parameter is used to specify one of the unique identifiers that refer to a particular object in the directory service. This lets you perform actions on a specific directory object, such as a particular user or group.

The primary unique identifier of an object is always a GUID—a 128-bit identifier, such as 7f5bfccd-fd08-49f5-809d-9ee2f9d7e845. This identifier never repeats and is therefore always unique. However, since a GUID is not easy to type, the *Identity* parameter also accepts values of other identifiers that are unique across a set of objects. Depending on the object you refer to, these could be the distinguished name (DN), security identifier (SID), user principal name (UPN), or pre-Windows 2000 user logon name or group name in the form Domain\Name.

The *Identity* parameter is also considered a positional parameter. The first argument on a cmdlet is assumed to be the *Identity* parameter when no parameter name is specified. This reduces the number of keystrokes when you type commands. For more information about positional parameters, see "Positional Parameters" earlier in this section.

Type of Identifier

When you specify a value for the *Identity* parameter, a cmdlet uses a certain heuristic process to determine the type of the identifier. To avoid ambiguities and improve performance, you can add a prefix to the parameter value in order to explicitly specify the type of the identifier:

```
-Identity '<prefix>=<identifier>'
```

These prefixes are also supported by other parameters that accept object identifiers as parameter values, such as the *SearchRoot*, *ParentContainer*, or *Member* parameter.

The following table lists the supported prefixes.

PREFIX	TREAT THE IDENTIFIER AS
upn	User principal name (UPN). Example: 'upn=user@domain'
dn	Distinguished name (DN). Example: 'dn=cn=user,dc=domain'
account	Pre-Windows 2000 logon name or group name in the form domain\name. Example: 'account=domain\user'
canonical	Canonical name in the form domain/container/…/name. Example: 'canonical=domain/users/user'
sid	Security identifier (SID). Example: 'sid=S-1-5-21-1216921794-1536856817-1513834708-1267'

guid	Globally unique identifier (GUID). Example: 'guid=4F881367-74A0-4CED-B9FB-25620A5D40ED'
anr	A value to be resolved using ambiguous name resolution (ANR). Normally, ANR supports the following attributes: • displayName • givenName (First Name) • sn (Last Name) • legacyExchangeDN • physicalDeliveryOfficeName (Office Location) • proxyAddresses • name (RDN) • sAMAccountName (pre-Windows 2000 logon name)

Syntax

ActiveRoles Management Shell follows the Windows PowerShell command conventions that help you understand what information is required or optional when you run a cmdlet and how you must present the parameters and their values. The following table lists these command conventions.

SYMBOL	DESCRIPTION
-	A hyphen indicates that the next word on the command line is a parameter. For more information about parameters, see "Parameters" earlier in this document.
< >	Angle brackets are used to indicate parameter values along with the parameter type setting. This setting specifies the form that the parameter's value should take, and refers to the .NET type that determines the kind of value that is permitted as a parameter argument. For example, <Int32> indicates that the parameter argument must be an integer; <String> indicates that the argument must be in the form of a character string. If the string contains spaces, the value must be enclosed in quotation marks or the spaces must be preceded by the escape character (`). The angle brackets are only intended to help you understand how a command should be constructed. You do not type these brackets when you enter the command on the command line.
[ ]	Square brackets are used to indicate an optional parameter and its value. A parameter and its value that are not enclosed in square brackets are required. If you do not supply a required parameter on the command line, the shell prompts you for that parameter. The square brackets are only intended to help you understand how a command should be constructed. You do not type these brackets when you enter the command on the command line.

In the Help documentation, all cmdlets display their associated parameters in parameter sets. These are groupings of parameters that can be used with each other. Although a cmdlet may have multiple parameter sets, most cmdlets have only one set of parameters. The following example displays the parameter set of the **Add-QADGroupMember** cmdlet:

```
Add-QADGroupMember [[-Identity] <String>] [-Proxy]
[-Service <String>] [ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<AdsiConnection>] -Member <String>
```

In this example:

- The name of the *Identity* parameter is enclosed in square brackets to indicate that you can specify the string value for this parameter without typing -Identity (this is a *positional parameter*, see "Parameters" earlier in this document).
- Since *Identity* is an optional parameter with this cmdlet, the [-Identity] <string> token is enclosed in square brackets.
- The *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, and *Connection* parameters along with their parameter values are enclosed in square brackets, to indicate that these are optional parameters, so each of these parameters along with their values can be omitted.
- *Member* is a required parameter, and thus it must be specified along with its string value, so the parameter name and value are not enclosed in square brackets.

Pipelining

The term *pipelining* refers to the act of having one cmdlet use the output of another cmdlet when it performs an operation. Pipelining is accomplished by using the pipe character (|). To create a pipeline, you connect cmdlets together with the pipe character. The result is that the output of the cmdlet preceding the pipe character is used as input to the cmdlet following the pipe character.

It is important to be aware that the shell does not pipe text between cmdlets. Instead, it pipes objects. From a user perspective, each object represents related information as a unit, making it easier to manipulate the information and extract specific pieces of information.

Thus, for bulk provisioning of user accounts by ActiveRoles Server based on data held in a text (CSV) file, you can run the following command (this command implies that the appropriate provisioning policies are configured in ActiveRoles Server to auto-populate the attributes, such as sAMAccountName, that are required for a user account to exist):

```
Import-Csv c:\temp\data.csv | ForEach-Object -Process
{New-QADUser -Proxy -ParentContainer 'OU=User,DC=company,DC=com' -Name $_.'user name'}
```

In this example:

- The **Import-Csv** cmdlet produces a set of objects, with each object representing one of the records found in the CSV file specified, and passes (pipes) the objects to the **ForEach-Object** cmdlet.
- The **ForEach-Object** cmdlet applies the specified script block to each of the incoming (piped) objects.
- For each of the incoming objects, the script block runs the **New-QADUser** cmdlet to create a user account with the name set to the value retrieved from the **user name** property of the object. The presence of -Proxy ensures that the operation is performed via ActiveRoles Server. The script accesses the incoming object through the **$_** variable provided by Windows PowerShell.

  Since the **user name** property value is the value found in the "user name" field of the CSV file record represented by the incoming object, the name of the newly created user account is appropriately set up based on the data retrieved from the CSV file.

Aliases

ActiveRoles Management Shell uses the aliasing mechanism provided by Windows PowerShell to assign concise aliases to cmdlet names and parameter names. An alias is an alternate, short name for a cmdlet or parameter. The native Windows PowerShell cmdlets have predefined, built-in aliases for cmdlet names. For example, **gcm** is an alias for **Get-Command**.

The cmdlets that come with ActiveRoles Management Shell do not have aliases for cmdlet names by default. The instructions on how to create an alias for a cmdlet name are given later in this section.

To list all cmdlet name aliases that are defined within your ActiveRoles Management Shell session, type the following command at the PowerShell command-prompt:

```
get-alias
```

To find the aliases for a cmdlet name, type:

```
get-alias | where-object {$_.definition -eq "<cmdlet-name>"}
```

For example, to find the aliases for **Get-Command**, type:

```
get-alias | where-object {$_.definition -eq "Get-Command"}
```

Aliases are helpful when you frequently use certain cmdlets and want to reduce the typing that you must do. When typing a command at the Windows PowerShell command-prompt, you can type aliases in place of cmdlet names and parameter names. For example, you may create the **gqu** alias for the **Get-QADUser** cmdlet and type **gqu** instead of typing **Get-QADUser** every time you need to use that cmdlet. You can create multiple aliases for the same cmdlet. Having aliases for a cmdlet does not prevent you from using the original name of the cmdlet.

Cmdlet parameters may also have aliases, in addition to parameter names. Parameter aliases are predefined and cannot be altered. Neither can you add your own, custom aliases for parameter names. To list all aliases defined for the parameter names specific to a certain cmdlet, type the following command at the Windows PowerShell command prompt:

```
gcm <cmdlet-name> | select -expand parametersets | select cmdname -expand parameters |
where {$_.aliases} | sort name | %{$lc="";$lp=""} {if ($lp -ne $_.Name) {$lp = $_.Name;
$_}} | ft name,aliases -auto
```

For example, to find the parameter aliases specific to the **Get-QADUser** cmdlet, type:

```
get-Command Get-QADUser | select -expand parametersets | select cmdname -expand
parameters | where {$_.aliases} | sort name | %{$lc="";$lp=""} {if ($lp -ne $_.Name) {$lp
= $_.Name; $_}} | ft name,aliases -auto
```

This command produces a two-column list (see the excerpt below), with parameter names listed in the first column. For each parameter name, the second column displays the alias (or aliases) that can be used in place of the parameter name. Thus, as shows the example below, when you want to use the *ConnectionAccount* parameter, you may type **User** or **ca**.

```
Name                Aliases
----                -------
AttributeScopeQuery {ASQ}
City                {l}
ConnectionAccount   {User, ca}
ConnectionPassword  {Pwd, cp}
Credential          {Cred}
Department          {dept}
```

```
DisplayName         {disp}
FirstName           {givenName, fn}
HomePhone           {hp}
Initials            {i}
LastName            {sn, ln}
LdapFilter          {lf}
Manager             {mgr}
MobilePhone         {mobile}
```

Creating an Alias for a Cmdlet Name

To create aliases for cmdlet names, use the **Set-Alias** cmdlet. For example, to create the **gqu** alias for **Get-QADUser**, type:

```
set-alias gqu get-qaduser
```

If you no longer need an alias, you can delete it by using the **Remove-Item** cmdlet to delete the alias from the **Alias:** drive. For example, to delete the **gqu** alias, type:

```
remove-item alias:gqu
```

Adding an Alias to a Windows PowerShell Profile

Aliases that are created from the command line by using the **Set-Alias** cmdlet during an ActiveRoles Management Shell session can be used when the session is active. After the session is closed, the alias definition is lost. To make your custom alias persistent and available every time that a new ActiveRoles Management Shell session is opened, you have to add the alias definition to your Windows PowerShell profile.

So, to retain your alias definitions, you should add the appropriate set-alias commands to a Windows PowerShell profile. The profile is loaded every time that Windows PowerShell starts.

To load a profile, your Windows PowerShell execution policy must permit you to load configuration files. If it does not, the attempt to load the profile fails and Windows PowerShell displays an error message.

The default execution policy, Restricted, does not permit any configuration files, including a Windows PowerShell profile, to be loaded. However, if you want to load configuration files, you can change the execution policy on your system. For information and instructions, type:

```
get-help about_signing
```

To see what the execution policy is in effect on your system, type:

```
get-executionpolicy
```

To change the execution policy on your system, use the **Set-ExecutionPolicy** cmdlet. For example, to enable the loading of Windows PowerShell profiles, change the execution policy to **RemoteSigned**. To do this, type the following command at the Windows PowerShell command-prompt:

```
set-executionpolicy remotesigned
```

Creating and Editing the Windows PowerShell User Profile

A Windows PowerShell user profile is not created automatically. The location of this profile is stored in the **$profile** variable, so you can determine if your user profile has been created by typing:

```
test-path $profile
```

If the profile exists, the response is **True**; otherwise, it is **False**.

To create your user profile, type:

```
new-item -path $profile -itemtype file -force
```

To open the profile in Notepad, type:

```
notepad $profile
```

Add the set-alias commands to the text in Notepad, one command per string (for example, **set-alias gqu get-qaduser**), save your changes (press Ctrl+S), and then close Notepad. Your alias definitions will be loaded every time that you open ActiveRoles Management Shell.

What's New in Version 1.2

ActiveRoles Server Version Compatibility

Version 1.2 of this solution is compatible with ActiveRoles Server of version 6.1 only. When a cmdlet attempts to connect to the ActiveRoles Server Administration Service of an earlier version, it receives an error message stating that the connection cannot be established because of version incompatibility.

Cmdlets

The following new cmdlets are available with the 1.2 release of this solution (see cmdlet descriptions in the cmdlet reference, later in this document):

- Get-QADMemberOf
- Get-QADPasswordSettingsObjectAppliesTo
- Restore-QADDeletedObject
- Get-QARSOperation
- Get-QARSApprovalTask
- Approve-QARSApprovalTask
- Reject-QARSApprovalTask

Parameters

The following table summarizes the parameters added on certain cmdlets with the 1.2 release of this solution (see parameter descriptions for the respective cmdlets in the cmdlet reference, later in this document).

PARAMETERS	ADDED ON
• Tombstone • LastKnownParent	• Get-QADComputer • Get-QADGroup • Get-QADObject • Get-QADPasswordSettingsObject • Get-QADUser
• CreatedOn • CreatedAfter • CreatedBefore • LastChangedOn • LastChangedAfter • LastChangedBefore	• Get-QADComputer • Get-QADGroup • Get-QADObject • Get-QADPasswordSettingsObject • Get-QADUser • Get-QARSAccessTemplate • Get-QARSAccessTemplateLink
• Type • ObjectAttributes • Name • DisplayName • Description • Anr	• Get-QADGroupMember
• MemberOf • IndirectMemberOf • NotMemberOf • NotIndirectMemberOf	• Get-QADComputer • Get-QADGroup • Get-QADObject • Get-QADUser
• ContainsMember • ContainsIndirectMember • NotContainsMember • NotContainsIndirectMember	• Get-QADGroup

Troubleshooting

In this section you can find information on some issues you may experience when using ActiveRoles Management Shell.

Script blocks in cmdlet parameter values may not work as expected

This issue applies only to the parameters that do not accept input from pipeline. Thus, the following syntax cannot be used to set a password value:

```
Get-QADUser Identity | Set-QADUser -Password {$_.SamAccountName}
```

An alternative syntax should be used in this case:

```
Get-QADUser Identity | %{Set-QADUser $_ -Password $_.SamAccountName}
```

However, you can use a script block to specify an identity, because the Identity parameter accepts input from pipeline:

```
Import-Csv c:\test.csv | Set-QADUser -Identity {$_.samAccountName} -Import
```

Not all membership-related parameters can be used in proxy mode

The following parameters cannot be used in conjunction with the Proxy parameter:

- ContainsIndirectMember
- NotContainsIndirectMember
- IndirectMemberOf
- NotIndirectMemberOf

If you attempt to use any of these parameters in the proxy mode, you receive an error.

Cmdlet Reference - Active Directory

Here you can find information about command-line tools (cmdlets) that are provided by ActiveRoles Management Shell.

This section covers the cmdlets for managing directory data, such as user or group properties. Supported are both Active Directory Domain Services and Active Directory Lightweight Directory Services.

Requirements on Active Directory

ActiveRoles Management Shell retains most of its features and functions when managing Windows 2000 Server based Active Directory. However, certain cmdlets and parameters require Active Directory of a later version:

- **Windows Server 2003** The cmdlets and parameters that rely on the attribute scope query (ASQ) search preference require the Active Directory functional level of Windows Server 2003 or higher. These include the **Get-QADGroupMember** cmdlet and the *AttributeScopeQuery* parameter.
- **Windows Server 2008** The cmdlets for managing Password Settings objects (such as **New-QADPasswordSettingsObject** or **Add-QADPasswordSettingsObjectAppliesTo**) require Windows Server 2008 based Active Directory.

Connect-QADService

Connect to the ActiveRoles Server Administration Service via the ActiveRoles Server ADSI Provider, or to a certain Active Directory domain controller or a certain server running an Active Directory Lightweight Directory Services (AD LDS) instance via the regular LDAP ADSI Provider.

Syntax

```
Connect-QADService [[-Service] <String>] [-Proxy] [-UseGlobalCatalog]
[-ConnectionAccount <String>] [-ConnectionPassword <SecureString>] [-Credential
<PSCredential>] [-Connection <ArsConnection>]
```

Parameters

Service

This is the fully qualified domain name, NetBIOS name or IP address of the computer running the Administration Service to connect to, or, if the Proxy parameter is not specified, the fully qualified domain name, NetBIOS name or IP address of the AD domain controller, or AD LDS server to connect to. In case of an AD LDS server, the fully qualified domain name of the server should be specified, with the appropriate port number added to the server name (see examples). If the DNS name of an AD domain is specified as the parameter value, then a connection is established to an appropriate domain controller in that domain.

With this parameter omitted, the Proxy parameter causes a connection to any available Administration Service. If both the Service and Proxy parameters are omitted, a connection is established to a domain controller in the domain of the computer running the cmdlet.

Proxy

If this parameter is present, the cmdlet will use the ActiveRoles Server ADSI Provider, so as to establish a connection using ActiveRoles Server. Otherwise, the regular Microsoft LDAP ADSI Provider will be used, so as to establish a direct connection to an AD domain or AD LDS server.

ConnectionAccount

This is the user logon name of the account with which you want to connect, in the form DomainName\UserName, or in the form of a user principal name. If this parameter is omitted, a connection is established with the credentials of the account under which the cmdlet is running.

ConnectionPassword

This is the password of the user account with which you want to connect. Use this parameter in conjunction with ConnectionAccount, to connect with the credentials of an account other than that under which the cmdlet is running.

The parameter value must be a SecureString object. Use the Read-Host cmdlet provided by Windows PowerShell to pass a SecureString object to this parameter.

Credential

This is the user name and password of the user account with which you want to connect, in the form of a PSCredential object. Use the Get-Credential cmdlet provided by Windows PowerShell to pass a PSCredential object to this parameter, if you want to connect with the credentials of an account other than that under which the cmdlet is running.

Connection

With this parameter, the credentials of an earlier established connection can be re-used to establish a new connection (for example, to a different server). Save in a certain variable the object returned by the Connect-QADService cmdlet, and then pass that object to this parameter when establishing a new connection.

UseGlobalCatalog

This parameter directs the cmdlet to connect to a domain controller that holds the role of the Global Catalog server. When the Proxy parameter is supplied, UseGlobalCatalog has no effect.

If UseGlobalCatalog is supplied together with the Service parameter that specifies a certain domain controller, the cmdlet connects to the specified domain controller if that domain controller is a Global Catalog server. If the Service parameter specifies a particular domain, then UseGlobalCatalog causes the cmdlet to connect to any available Global Catalog server in that domain. If the Service parameter is omitted, then UseGlobalCatalog causes the cmdlet to connect to any available Global Catalog server in the domain of the computer running the cmdlet.

Detailed Description

This cmdlet establishes a connection to any available Administration Service, to a specific Administration Service, or directly to a specific Active Directory domain controller or a server running an Active Directory Lightweight Directory Services (AD LDS) instance, with the credentials of the locally logged on user or with the credentials of a specified user. A connection determines the default connection parameters (the server and the security context) for the operations that are performed by the other cmdlets. The default connection parameters are effective until the connection is closed either explicitly or by establishing a new connection, and can be overridden on a per-cmdlet basis.

The cmdlet establishes a connection in the security context of a certain user, so some user credentials must be provided in order to authenticate the user.

The cmdlet makes it possible to specify user credentials in a number of ways through the use of the credential-related parameters *ConnectionAccount*, *ConnectionPassword*, *Credential*, and *Connection*:

- If no connection-related parameters are specified, the cmdlet uses the credentials of the locally logged on user.
- If the *Credential* parameter is specified, the credentials provided by this parameter are used regardless of whether any other credential-related parameters are specified.
- If the *ConnectionAccount* and *ConnectionPassword* parameters are specified while the *Credential* parameter is omitted, the specified user name and password are passed to the cmdlet as the user credentials regardless of whether the *Connection* parameter is specified.
- If the *Connection* parameter is specified while all the other credential-related parameters are omitted, the cmdlet re-uses the credentials that were used to open the existing connection.

The object that is returned by this cmdlet can be passed as the value of the *Connection* parameter to any other cmdlet in this snap-in in order to re-use the connection parameters of the existing connection. Note that the object includes information not only about the user credentials or security context, but also about the server to which the connection is established. So, if you pass the object to a cmdlet and omit the *Service* parameter, the cmdlet will use the server specified by the object you have passed to the cmdlet.

Examples

Example 1

Connect to any available domain controller with the credentials of the locally logged on user:

```
C:\PS> connect-QADService
```

Example 2

Connect to the local Administration Service with the credentials of the locally logged on user:

```
C:\PS> connect-QADService -service 'localhost' -proxy
```

Example 3

Prompt the user for password within the console window (in text mode); then, connect to a specific domain controller with the user name and password specified:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-QADService -service 'server.company.com' -ConnectionAccount
'company\administrator' -ConnectionPassword $pw
```

Example 4

Use a dialog box to request a user name and password; then, connect to a specific domain controller with those user name and password, and save the AdsiConnection object in a variable for later use:

```
C:\PS> $cred = get-credential
C:\PS> $conn = connect-QADService -service 'server.company.com' -credential $cred
```

Example 5

Connect to the AD LDS instance on 'server.domain.local:389' with the credentials of the locally logged on user:

```
C:\PS> connect-QADService -service 'server.domain.local:389'
```

Disconnect-QADService

Close the connection, if any exists. A connection could be established by using the **Connect-QADService** cmdlet.

Syntax

```
Disconnect-QADService [-Proxy] [-UseGlobalCatalog] [-Service <String>]
[-ConnectionAccount <String>] [-ConnectionPassword <SecureString>] [-Credential
<PSCredential>] [-Connection <ArsConnection>]
```

Parameters

This cmdlet takes the same optional connection parameters as the **Connect-QADService** cmdlet. The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see "Parameters" in the "Connect-QADService" section earlier in this document.

Detailed Description

Any connection established using the **Connect-QADService** cmdlet must be finally closed by executing the **Disconnect-QADService** cmdlet. The cmdlet closes the last open connection, if any. If the *Connection* parameter is present, the cmdlet also closes the connection specified by the value of that parameter. If no connection is currently open, the cmdlet attempts to establish a connection in accordance with the connection parameters specified, and then closes the connection.

Examples

Example 1

Close the last open connection, if any:

```
C:\PS> disconnect-QADService
```

Example 2

Close the last open connection and also close the connection defined by an AdsiConnection object that was earlier saved in the $conn variable:

```
C:\PS> disconnect-QADService -connection $conn
```

Get-QADUser

Retrieve all users in a domain or container that match the specified conditions. Supported are both Active Directory Domain Services (AD DS) and Active Directory Lightweight Directory Services (AD LDS).

Syntax

```
Get-QADUser [[-Identity] <IdentityParameter>] [-Manager <IdentityParameter>] [-City
<String>] [-Company <String>] [-Department <String>] [-Fax <String>] [-FirstName
<String>] [-HomePhone <String>] [-Initials <String>] [-LastName <String>] [-MobilePhone
<String>] [-Notes <String>] [-Office <String>] [-Pager <String>] [-PhoneNumber <String>]
[-PostalCode <String>] [-PostOfficeBox <String>] [-SamAccountName <String>]
[-StateOrProvince <String>] [-StreetAddress <String>] [-Title <String>]
[-UserPrincipalName <String>] [-WebPage <String>] [-HomeDirectory <String>] [-HomeDrive
<String>] [-ProfilePath <String>] [-LogonScript <String>] [-Email <String>] [-Disabled]
[-Enabled] [-Locked] [-AccountExpiresBefore <DateTime>] [-AccountExpiresAfter
<DateTime>] [-AccountNeverExpires] [-PasswordNeverExpires] [-MemberOf
<IdentityParameter[]>] [-IndirectMemberOf <IdentityParameter[]>] [-NotMemberOf
<IdentityParameter[]>] [-NotIndirectMemberOf <IdentityParameter[]>] [-Tombstone]
[-LastKnownParent <IdentityParameter>] [-SecurityMask <SecurityMasks>] [-SearchRoot
<IdentityParameter>] [-SearchScope <SearchScope>] [-AttributeScopeQuery <String>]
[-PageSize <Int32>] [-SizeLimit <Int32>] [-LdapFilter <String>] [-WildcardMode
<WildcardMode>] [-ObjectAttributes <Object>] [-Description <String>] [-DisplayName
<String>] [-Name <String>] [-Anr <String>] [-CreatedOn <DateTime>] [-CreatedAfter
<DateTime>] [-CreatedBefore <DateTime>] [-LastChangedOn <DateTime>] [-LastChangedAfter
<DateTime>] [-LastChangedBefore <DateTime>] [-IncludeAllProperties]
[-DontConvertValuesToFriendlyRepresentation] [-SerializeValues]
[-ReturnPropertyNamesOnly] [-DontUseDefaultIncludedProperties]
[-UseDefaultExcludedProperties] [-ExcludedProperties <String[]>] [-IncludedProperties
<String[]>] [-UseDefaultExcludedPropertiesExcept <String[]>] [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>]
```

The cmdlet has a number of optional parameters for searching by individual attributes in the directory, with each parameter name identifying a certain attribute that you can search for attribute values specified by using the respective parameter (see the list of parameters for this cmdlet).

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

AccountExpiresAfter

Retrieve user accounts that are configured to expire after a certain date. Parameter value is a DateTime object that specifies the date you want.

AccountExpiresBefore

Retrieve user accounts that are configured to expire before a certain date. Parameter value is a DateTime object that specifies the date you want.

AccountNeverExpires

Set this parameter to 'true' for the cmdlet to retrieve the accounts configured to never expire.

Anr

Specify a value to be resolved using ambiguous name resolution (ANR). Which attributes are included in an ANR search depends upon the Active Directory schema. Thus, in Windows Server 2003 based Active Directory, the following attributes are set for ANR by default:

- Display-Name (displayName)
- Given-Name (givenName)
- Legacy-Exchange-DN (legacyExchangeDN)
- ms-DS-Additional-Sam-Account-Name (msDS-AdditionalSamAccountName)
- Physical-Delivery-Office-Name (physicalDeliveryOfficeName)
- Proxy-Addresses (proxyAddresses)
- RDN (name)
- SAM-Account-Name (sAMAccountName)
- Surname (sn)

For instance, when you supply 'ann*' as the value of this parameter, the cmdlet searches for objects that have ann at the beginning of the value of at least one of the attributes listed above.

AttributeScopeQuery

Specify the LDAP display name of an attribute that has DN syntax (for example, "member" or "memberOf"). The cmdlet enumerates the Distinguished Name values of the attribute on the object specified by the SearchRoot parameter, and performs the search on the objects represented by the Distinguished Names. The SearchScope parameter has no effect in this case. The object to search must be specified by using the SearchRoot parameter rather than the Identity parameter.

For instance, with the value of this parameter set to "memberOf", the cmdlet searches the collection of the groups to which the SearchRoot object belongs.

CreatedAfter

Specify the lower boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created after the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedBefore

Specify the upper boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created before the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedOn

Specify the object creation date by which to filter objects found, searching for objects created within the date specified. This parameter is mutually exclusive with the CreatedAfter and CreatedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

Disabled

Supply this parameter for the cmdlet to search for disabled accounts.

DontConvertValuesToFriendlyRepresentation

This parameter causes the cmdlet to represent the Integer8 and OctetString attribute values "as is," without converting them to a user-friendly, human-readable form. If this parameter is omitted, the cmdlet performs the following data conversions:

- The values of the Integer8 attributes listed in the Integer8AttributesThatContainDateTimes array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to DateTime.
- The values of the Integer8 attributes listed in the Integer8AttributesThatContainNegativeTimeSpans array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to TimeSpan.
- The values of the other Integer8 attributes are converted from IADsLargeInteger to Int64.
- The values of the OctetString attributes are converted from byte[] to BinHex strings.

Note: This parameter has an effect only on the properties of the output object that have the member type of NoteProperty. Such properties are normally added to the output object in order to provide access to the attribute values of the respective directory object that are loaded to the local memory cache but cannot be accessed by using properties of the base object (the object for which the output object serves as a wrapper).

DontUseDefaultIncludedProperties

This parameter causes the cmdlet to load only a small set of attributes from the directory to the local memory cache (normally, this set is limited to objectClass and ADsPath). Other attributes are retrieved from the directory as needed when you use the cmdlet output objects to read attribute values. Thus, if you want only to count the objects that meet certain conditions (rather than examine values of particular attributes), then you can use this parameter to increase performance of your search. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

Enabled

Supply this parameter for the cmdlet to retrieve only those accounts that are enabled (not disabled).

ExcludedProperties

Use this parameter to specify the attributes that you do not want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the ExcludedProperties parameter you can change this default behavior on an ad-hoc basis, in order to prevent certain attributes from being loaded. Another scenario involves the use of this parameter in conjunction with IncludeAllProperties in order to restrict the set of the cached attributes.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to find.

The cmdlet attempts to retrieve the object identified by this parameter value, disregarding the other search parameters. If you want other search parameters to have effect, do not supply this parameter.

IncludeAllProperties

With this parameter, the cmdlet retrieves all attributes of the respective directory object (such as a User object), and stores the attribute values in the memory cache on the local computer. Attribute values can be read from the memory cache by using properties of the object returned by the cmdlet. Thus, when used in conjunction with the SerializeValues parameter, it allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

IncludedProperties

Use this parameter to specify the attributes that you want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the IncludedProperty parameter you can direct the cmdlet to cache some attributes in addition to the default set.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet.

IndirectMemberOf

Retrieve objects that belong to the group or groups specified by this parameter, whether directly or because of group nesting. The cmdlet returns an object if the object has direct or indirect membership in the group specified by this parameter value.

LastChangedAfter

Specify the lower boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed after the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedBefore

Specify the upper boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed before the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedOn

Specify the object modification date by which to filter objects found, searching for objects that have last changed within the date specified. This parameter is mutually exclusive with the LastChangedAfter and LastChangedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

LastKnownParent

When searching for a deleted object by using the Tombstone parameter, specify the DN of the container the object was in before it became a tombstone. This allows you to find objects that were deleted from a particular container.

Note that the lastKnownParent attribute is only set if the object was deleted on a domain controller running Windows Server 2003 or later version of Microsoft Windows Server. Therefore, it is possible that the lastKnownParent attribute value is inaccurate.

LdapFilter

Specify the LDAP search filter that defines your search criteria. Note that the search filter string is case-sensitive.

The cmdlet disregards this parameter if an Identity value is supplied. If you want this parameter to have effect, do not supply an Identity parameter value.

If you supply the LdapFilter parameter along with attribute-specific parameters, then your search returns objects that meet the conditions defined by the LDAP filter and have the specified attributes set to the specified values.

Locked

Supply this parameter for the cmdlet to retrieve only those accounts that are currently in the "locked" state.

MemberOf

Retrieve objects that are direct members of the group or groups specified by this parameter. The cmdlet returns an object if the object has direct membership in the group specified by this parameter value.

NotIndirectMemberOf

Retrieve objects that do not belong to the group or groups specified by this parameter, whether directly or because of group nesting. The cmdlet returns an object if the object has neither direct nor indirect membership in the group specified by this parameter value.

NotMemberOf

Retrieve objects that are not direct members of the group or groups specified by this parameter. The cmdlet returns an object if the object does not have direct membership in the group specified by this parameter value.

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet searches for objects that have the specified attributes set to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to search. A value may include an asterisk character - a wildcard representing any group of characters. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

PageSize

Set the maximum number of items in each page of the search results that will be returned by the cmdlet. After the directory server has found the number of objects that are specified by this parameter, it will stop searching and return the results to the cmdlet. When the cmdlet requests more data, the server will restart the search where it left off. You can use this setting to adjust the number of requests (network calls) to the directory server issued by the cmdlet during a search.

Normally, the default page size is 50. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

PasswordNeverExpires

Set this parameter to 'true' for the cmdlet to retrieve only those accounts that have the password configured to never expire.

ReturnPropertyNamesOnly

This parameter causes the cmdlet to list the names of the object attributes whose values the cmdlet retrieves from the directory and stores in the memory cache on the local computer. Thus, when used in conjunction with the IncludeAllProperties parameter, it lists the names of all attributes of the respective directory object (such as a User object). For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

SearchRoot

Specify the Distinguished Name, Canonical Name, or GUID of the container to search. By default, the cmdlet searches the entire sub-tree of which SearchRoot is the topmost object (sub-tree search). This default behavior can be altered by using the SearchScope parameter.

The search criteria are defined by the LdapFilter parameter value and the values of attribute-specific parameters.

The cmdlet disregards this parameter if an Identity value is supplied. If you want this parameter to have effect, do not supply any Identity parameter value.

SearchScope

Specify one of these parameter values:

- 'Base' - Limits the search to the base (SearchRoot) object. The result contains a maximum of one object.
- 'OneLevel' - Searches the immediate descendant (child) objects of the base (SearchRoot) object, excluding the base object.
- 'Subtree' - Searches the whole sub-tree, including the base (SearchRoot) object and all its descendant objects.

Normally, if this parameter is not supplied, the cmdlet performs a Subtree search. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

SecurityMask

Specify which elements of the object's security descriptor to retrieve. Valid parameter values are:

- 'None' - do not retrieve any security data
- 'Owner' - retrieve the owner data
- 'Group' - retrieve the primary group data
- 'Dacl' - retrieve the discretionary access-control list data
- 'Sacl' - retrieve the system access-control list data

You can supply a combination of these values, separating them by commas. For example, you can supply the parameter value of 'Dacl,Sacl' in order to retrieve both the discretionary and system access-control list data.

SerializeValues

This parameter causes the cmdlet to output an object whose properties store the attribute values of the respective directory object that are loaded to the local memory cache. The value returned by each property of the output object is represented as a string (serialized) so as to facilitate the export of the attribute values to a text file. Thus, when used in conjunction with the IncludeAllProperties parameter, this parameter allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

SizeLimit

Set the maximum number of items to be returned by the cmdlet. Normally, the default size limit is 1000. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

Tombstone

Search for deleted objects of the respective object class. The search output is normally intended to be passed (piped in) to the Restore-QADDeletedObject cmdlet for restoring deleted objects.

When Active Directory deletes an object from the directory, the object becomes a tombstone. A search using this parameter returns tombstone objects that meet the filtering criteria supplied. Note that upon deletion of an object only a small number of the object's attributes are saved in the tombstone, with most of the attributes being lost. To search for deleted objects, your search filter should be based on the attributes that are preserved in tombstones. Which attributes are preserved is determined by Active Directory. Thus, the attributes that have the 0x8 bit set in the searchFlags attribute of the attributeSchema definition are preserved.

When the Tombstone parameter is supplied, the search results include deleted objects (tombstones) that match the specified search filter. However, a search filter that matches a live object may not work as expected after the object is deleted. This is because not all attributes are retained in the tombstone. For example, a filter such as (&(objectClass=user)(objectCategory=person)) would not match any tombstone objects since the objectCategory attribute is removed upon object deletion. Conversely, the objectClass attribute is retained on tombstone objects, so a filter of (objectClass=user) would match deleted user objects.

The name (RDN) of a tombstone object begins with the name (RDN) of the deleted object, so a search using the Tombstone parameter can be refined by adding a filter based on object name. For example, to search for deleted objects with a name that begins with "John", you can use a filter such as (cn=John*).

It is also possible to find a specific deleted object. If you know the name (RDN) of the object and the Distinguished Name (DN) of the container the object was in before it was deleted, then you can pass the container's DN to the LastKnownParent parameter and apply a filter of (cn=<RDN of the object>*) in order to have the cmdlet retrieve that specific object. However, if an object is deleted, a new object with the same DN is created, and then deleted as well, the above search would return more than one tombstone object. The returned tombstone objects are distinguished by the GUIDs of the deleted objects, with the name of each tombstone object ending in the GUID of the respective deleted object.

UseDefaultExcludedProperties

When set to 'true', this parameter causes the cmdlet not to load a certain pre-defined set of attributes from the directory to the local memory cache. This pre-defined set of attributes (referred to as "default excluded properties") can be viewed or modified by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Normally, this parameter is used in conjunction with IncudeAllProperties to avoid retrieval of unnecessary data from the directory server, thereby increasing performance of the search operation performed by the cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

UseDefaultExcludedPropertiesExcept

This parameter is deprecated, and has no effect.

WildcardMode

Specify either 'PowerShell' or 'LDAP' as the parameter value. Normally, if this parameter is not supplied, the cmdlet assumes that WildcardMode is set to 'LDAP'. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. The 'PowerShell' value causes the cmdlet to use PowerShell wildcards and quoting rules. Wildcards are processed on the client side, which may result in slow search performance.

For information about PowerShell wildcards and quoting rules, type the following commands at the PowerShell command-prompt:

```
help about_wildcard
help about_quoting_rule
```

The 'LDAP' value causes the cmdlet to use LDAP wildcards (asterisks only) and LDAP quoting rules (backslash as the escape character). Wildcards are processed on the server side, which enables faster search results.

Attribute-specific Parameters

The cmdlet takes a series of optional, attribute-specific parameters allowing you to search by user attributes. The attribute-specific parameters have effect if SearchRoot is specified whereas Identity is not. If you specify SearchRoot only, then the cmdlet returns all users found in the SearchRoot container.

You can use attribute-specific parameters to search for user accounts that have specific values of certain attributes. Thus, to find all user accounts that have the "givenName" attribute set to Martin, you may add the following on the command line: -FirstName Martin. To search for user accounts that have a certain attribute not set specify '' (empty string) as the parameter value.

If a particular attribute is referred to by both the ObjectAttributes array and the attribute-specific parameter, the ObjectAttributes setting has no effect on that attribute. The cmdlet searches for the attribute value specified by the attribute-specific parameter.

With more than one attribute-specific parameter supplied, the search conditions are combined by using the AND operator, so as to find the user accounts that meet all the specified conditions. Thus, if you supply both the *FirstName* and *LastName* parameters, the cmdlet searches for the user accounts that have the "givenName" attribute set to the FirstName parameter value and the "sn" attribute set to the LastName parameter value.

Each of these parameters accepts the asterisk (*) wildcard character in the parameter value to match zero or more characters (case-insensitive). For instance, a* matches A, ag, Amsterdam, and does not match New York.

The following table lists the attribute-specific parameters you can use with this cmdlet. Each parameter adds a filter condition based on a certain attribute identified by the LDAP display name in the table.

TO SEARCH BY THIS ATTRIBUTE...	USE THIS SYNTAX
l	-City <String>
company	-Company <String>
description	-Description <String>
department	-Department <String>

displayName	-DisplayName <String>
facsimileTelephoneNumber	-Fax <String>
givenName	-FirstName <String>
homeDirectory	-HomeDirectory <String>
homeDrive	-HomeDrive <String>
homePhone	-HomePhone <String>
initials	-Initials <String>
sn	-LastName <String>
mail	-Email <String>
manager	-Manager <IdentityParameter>
mobile	-MobilePhone <String>
name	-Name <String>
info	-Notes <String>
physicalDeliveryOfficeName	-Office <String>
pager	-Pager <String>
telephoneNumber	-Phone <String>
postalCode	-PostalCode <String>
postOfficeBox	-PostOfficeBox <String>
profilePath	-ProfilePath <String>
samAccountName	-SamAccountName <String>
scriptPath	-LogonScript <String>
st	-StateOrProvince <String>
streetAddress	-StreetAddress <String>
title	-Title <String>
userPrincipalName	-UserPrincipalName <String>
wWWHomePage	-WebPage <String>

Detailed Description

Use this cmdlet to search an Active Directory domain or container for user accounts that meet certain criteria, or to bind to a certain user account by DN, SID, GUID, UPN, or Domain\Name. You can search by user attributes or specify your search criteria by using an LDAP search filter.

The output of the cmdlet is a collection of objects, with each object representing one of the user accounts found by the cmdlet. You can pipe the output into another cmdlet, such as **Set-QADUser**, to make changes to the user accounts returned by this cmdlet.

Examples

Example 1

Connect to any available domain controller with the credentials of the locally logged on user, bind to a specific user account by Domain\Name, and display the user description. In this example, the NetBIOS name of the domain is assumed to be "MyDomain" and the pre-Windows 2000 name of the user account is assumed to be "MyLogonName":

```
C:\PS> (get-QADUser 'MyDomain\MyLogonName').DirectoryEntry.description
```

Example 2

Connect to a specific domain controller with the credentials of a specific user, bind to a certain user account by SID, display the user description, and then disconnect:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-QADService -service 'server.company.com' -ConnectionAccount
'company\administrator' -ConnectionPassword $pw
C:\PS> (get-QADUser -identity
'S-1-5-21-1279736177-1630491018-182859109-1305').DirectoryEntry.description
C:\PS> disconnect-QADService
```

Example 3

Connect to any available domain controller with the credentials of the locally logged on user, search for users in a specific container by using an LDAP search filter, and display a list of the users found:

```
C:\PS> get-QADUser -SearchRoot 'company.com/UsersOU' -LdapFilter '(description=a*)'
```

Example 4

Connect to any available domain controller with the credentials of the locally logged on user, find all users in a specific container, and display a list of the users found:

```
C:\PS> get-QADUser -SearchRoot 'company.com/UsersOU'
```

Example 5

Connect to any available domain controller with the credentials of a specific user, search a certain container to find all users with empty title, set a title for each of those users, and then disconnect:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-QADService -ConnectionAccount 'company\administrator'
-ConnectionPassword $pw
C:\PS> get-QADUser -SearchRoot 'company.com/UsersOU' -title '' | set-QADUser -title
'Contractor'
C:\PS> disconnect-QADService
```

Example 6

Connect to the local Administration Service with the credentials of the locally logged on user, find all users whose names begin with "A" and titles end in "Manager" and modify the description for each of those users; then, disconnect.

```
C:\PS> connect-QADService -service 'localhost' -proxy
C:\PS> get-QADUser -SearchRoot 'company.com/UsersOU' -Name 'A*' -ObjectAttributes
@{name='B*';title='*manager'} | set-QADUser -description 'A manager whose name begins
with A'
C:\PS> disconnect-QADService
```

Note that the condition based on the *Name* parameter overrides the condition imposed on the "Name" attribute by the *ObjectAttributes* parameter, so you could omit the *Name* parameter and type name='A*' instead of name='B*' in the value of the *ObjectAttributes* parameter, or you could only remove the name='B*' entry from the value of the *ObjectAttributes* parameter.

Example 7

List the names of the properties specific to a user object:

```
C:\PS> Get-QADUser -IncludeAllProperties -ReturnPropertyNamesOnly
```

Example 8

List the values of all properties of the user account:

```
C:\PS> Get-QADUser JSmith -IncludeAllProperties -SerializeValues | Format-List
```

Example 9

Export the user account to an XML file. Exported are the values of all properties:

```
C:\PS> Get-QADUser jsmith -IncludeAllProperties -SerializeValues | Export-Clixml
user.xml
```

Example 10

Find user objects with a non-empty value of the 'homeDirectory' property, and display the values of the 'Name', 'HomeDirectory' and 'msDS-ReplAttributeMetaData' properties for each object found:

```
C:\PS> Get-QADUser -DontUseDefaultIncludedProperties -ObjectAttributes
@{homeDirectory='*'} -IncludedProperties 'msDS-ReplAttributeMetaData',homeDirectory |
Format-Table name, homeDirectory, 'msDS-ReplAttributeMetaData'
```

Example 11

Export the user object to a CSV file. Then, import that user object from that file:

```
C:\PS> Get-QADuser jsmith -SerializeValues | export-csv user.csv
C:\PS> import-csv user.csv | New-QADUser -ParentContainer 'MyDomain.lab.local/MyOU'
-DeserializeValues -Name importedUser -LogonName importedUser -UserPassword 'P@ssw0rd'
```

Example 12

Count all user objects that exist in your Active Directory domain:

```
C:\PS> Get-QADUser -DontUseDefaultIncludedProperties -SizeLimit 0 | Measure-Object
```

Example 13

Connect to the AD LDS instance on 'server.domain.local:389' with the credentials of the locally logged on user, search a specific container to find all AD LDS user objects matching a certain LDAP search filter, and display the name and description of each user object found:

```
C:\PS> get-QADUser -Service 'server.domain.local:389' -SearchRoot '<DN of container>'
-LdapFilter '(description=a*)' | Format-List name,description
```

Example 14

Connect to the AD LDS instance on 'server.domain.local:389' with the credentials of the locally logged on user, and display the name and description of the AD LDS user object that is identified by DN:

```
C:\PS> get-QADUser '<DN of user object>' -Service 'server.domain.local:389' | Format-List
name,description
```

Example 15

Retrieve user accounts from a particular container that are direct or indirect members of a particular group:

```
C:\PS> get-QADUser -SearchRoot '<DN of container>' -IndirectMemberOf
'domainName\groupName'
```

Example 16

Retrieve all user accounts that were deleted from a particular container:

```
C:\PS> Get-QADUser -Tombstone -LastKnownParent '<DN of container>'
```

Example 17

Retrieve deleted user accounts with the name (RDN) of John Smith:

```
C:\PS> Get-QADUser -Tombstone -Name 'John Smith*'
```

Example 18

Retrieve all user accounts that were deleted from a particular container on the current date:

```
C:\PS> Get-QADUser -Tombstone -LastKnownParent '<DN of container>' -LastChangedOn
(get-date)
```

Example 19

Retrieve all user accounts that were deleted on September 1, 2008:

```
C:\PS> Get-QADUser –Tombstone –LastChangedOn (get-date -year 2008 -month 9 -day 1)
```

Return Object Properties

Properties and methods of the object returned by the Get-QADUser cmdlet can be used to examine and configure various properties of the respective user account. To view a list of all methods and properties that are available, use the following command:

```
get-QADUser 'domainname\username' | get-Member
```

For general information about using properties and methods of PowerShell objects, enter these commands:

```
get-help about_method
get-help about_property
```

The following table summarizes some properties of a Get-QADUser return object. Using these properties you can view or modify properties on a user account you retrieve with the cmdlet. After setting new property values on the cmdlet return object, you must call the CommitChanges() method on that object to save the property value changes in the user account (see examples at the end of this section).

PROPERTY	DESCRIPTION
AccountExpires Syntax: Nullable<DateTime>	The date and time after which the user cannot log on.
AccountIsDisabled Syntax: Boolean	A flag to indicate if the account is, or should be, disabled.
AccountIsLockedOut Syntax: Boolean	A flag that indicates if the account is locked because of failed logon attempts.
Department Syntax: String	The department within the company to which the user belongs.
Description Syntax: String	The text description of the user.
Email Syntax: String	The e-mail address of the user.
Fax Syntax: String	The fax number of the user.
FirstName Syntax: String	The first name of the user.
HomeDirectory Syntax: String	The home directory of the user.
HomeDrive Syntax: String	The drive letter to which the UNC path for the home directory is mapped.
LastLogon Syntax: Nullable<DateTime>	The date and time that the user last logged on using the domain controller from which the user account is retrieved by the cmdlet.
LastLogonTimestamp Syntax: Nullable<DateTime>	The date and time that the user last logged on to the domain.
LastName Syntax: String	The last name of the user.
LogonScript Syntax: String	The logon script path.

Manager Syntax: String	Identifies the account of the user's manager.
MemberOf Syntax: String[]	Array of strings, each of which identifies one of the groups that the user is a member of.
PasswordAge Syntax: Nullable<TimeSpan>	Time that has elapsed since the password was set or last changed.
PasswordExpires Syntax: Nullable<DateTime>	The date and time when the password expires.
PasswordLastSet Syntax: Nullable<DateTime>	The date and time when the password was set or last changed.
PasswordNeverExpires Syntax: Boolean	A flag indicating if the password is configured to never expire.
ProfilePath Syntax: String	The path to the user profile.
TSAllowLogon Syntax: Boolean	A flag indicating if the user is allowed to log on to Terminal Services.
TSBrokenConnectionAction Syntax: Int32	The action to take when a Terminal Services session limit is reached: **1** if the session should be terminated; **0** if the session should be disconnected.
TSConnectClientDrives Syntax: Boolean	A flag indicating whether to reconnect to mapped client drives at logon to Terminal Services.
TSConnectPrinterDrives Syntax: Boolean	A flag indicating whether to reconnect to mapped client printers at logon to Terminal Services.
TSDefaultToMainPrinter Syntax: Boolean	A flag indicating whether to print automatically to the client's default printer when the user is logged on to the Terminal Server.
TSHomeDirectory Syntax: String	The Terminal Services home directory of the user.
TSHomeDrive Syntax: String	The drive letter to which the UNC path for the Terminal Services home directory is mapped.
TSInitialProgram Syntax: String	The path and file name of the application that starts automatically when the user logs on to Terminal Services.
TSMaxConnectionTime Syntax: TimeSpan	Maximum allowed duration of the Terminal Services session.
TSMaxDisconnectionTime Syntax: TimeSpan	Maximum amount of time that a disconnected Terminal Services session remains active on the Terminal Server.
TSMaxIdleTime Syntax: TimeSpan	Maximum amount of time that the Terminal Services session can remain idle.

TSProfilePath Syntax: String	The profile path to use when the user logs on to Terminal Services.
TSReconnectionAction Syntax: Int32	Specifies whether to allow reconnection to a disconnected Terminal Services session from any client computer: **1** if reconnection is allowed from the original client computer only; **0** if reconnection from any client computer is allowed.
TSRemoteControl Syntax: Int32	Specifies whether to allow remote observation or remote control of the user's Terminal Services session: • **0** Remote control is disabled. • **1** Full control of the user's session, with the user's permission. • **2** Full control of the user's session; the user's permission is not required. • **3** View the session remotely, with the user's permission. • **4** View the session remotely; the user's permission is not required.
TSWorkDirectory Syntax: String	The working directory path to use when the user logs on to Terminal Services.
UserMustChangePassword Syntax: Boolean	A flag indicating if the user is required to change the password at next logon.

Examples

Example 1

Force a particular user to change the password at next logon:

```
C:\PS> $user = get-QADUser 'DomainName\UserName'
C:\PS> ($user).UserMustChangePassword = $true
C:\PS> ($user).CommitChanges()
```

Example 2

View the **TSAllowLogon** setting on a specific user account:

```
C:\PS> (get-QADUser 'DomainName\AccountName').TSAllowLogon
```

Example 3

Set the **TSMaxIdleTime** property on a specific user account to 15 minutes; then, view the setting:

```
C:\PS> $user = get-QADUser 'DomainName\UserName'
C:\PS> ($user).TSMaxIdleTime = [TimeSpan]("0:15:0")
C:\PS> ($user).CommitChanges()
C:\PS> ($user).TSMaxIdleTime
```

Set-QADUser

Modify attributes of a user account in Active Directory. Supported are both Active Directory Domain Services (AD DS) and Active Directory Lightweight Directory Services (AD LDS).

Syntax

```
Set-QADUser [-Identity] <IdentityParameter> [-HomeDirectory <String>] [-HomeDrive
<String>] [-ProfilePath <String>] [-LogonScript <String>] [-AccountExpires <Nullable`1>]
[-PasswordNeverExpires] [-UserMustChangePassword] [-Email <String>] [-TsProfilePath
<String>] [-TsHomeDirectory <String>] [-TsHomeDrive <String>] [-TsWorkDirectory
<String>] [-TsInitialProgram <String>] [-TsMaxDisconnectionTime <TimeSpan>]
[-TsMaxConnectionTime <TimeSpan>] [-TsMaxIdleTime <TimeSpan>] [-TsAllowLogon]
[-TsRemoteControl <Int32>] [-TsReconnectionAction <Int32>] [-TsBrokenConnectionAction
<Int32>] [-TsConnectClientDrives] [-TsConnectPrinterDrives] [-TsDefaultToMainPrinter]
[-City <String>] [-Company <String>] [-Department <String>] [-Fax <String>] [-FirstName
<String>] [-HomePhone <String>] [-Initials <String>] [-LastName <String>] [-Manager
<IdentityParameter>] [-MobilePhone <String>] [-Notes <String>] [-Office <String>]
[-Pager <String>] [-PhoneNumber <String>] [-PostalCode <String>] [-PostOfficeBox
<String>] [-SamAccountName <String>] [-StateOrProvince <String>] [-StreetAddress
<String>] [-Title <String>] [-UserPrincipalName <String>] [-WebPage <String>]
[-UserPassword <String>] [-ObjectAttributes <ObjectAttributesParameter>] [-Description
<String>] [-DisplayName <String>] [-ExcludedProperties <String[]>] [-IncludedProperties
<String[]>] [-DeserializeValues] [-UseDefaultExcludedProperties]
[-UseDefaultExcludedPropertiesExcept <String[]>] [-Proxy] [-UseGlobalCatalog] [-Service
<String>] [-ConnectionAccount <String>] [-ConnectionPassword <SecureString>]
[-Credential <PSCredential>] [-Connection <ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has a number of optional parameters for managing individual attributes in the directory, with each parameter name identifying a certain attribute that can be set to a value specified by using the respective parameter (see the list of parameters for this cmdlet).

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

AccountExpires

Set the account expiration date. Parameter value is a DateTime object that specifies the date you want. A null DateTime object configures the account to never expire.

Confirm

Prompts you for confirmation before executing the command.

DeserializeValues

Supply this parameter if the input you pass to the cmdlet contains serialized attribute values (for instance, when importing a directory object from a text file that was created using the Serialize parameter). For examples of how to export and import an object, see documentation on the Get-QADUser cmdlet.

ExcludedProperties

Use this parameter to specify the attributes that you do not want the cmdlet to set in the directory. Supply a list of the attribute LDAP display names as the parameter value. You could use this parameter when importing attribute values from a text file, in order to prevent some attributes found in the file from being set in the directory.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

IncludedProperties

Use this parameter to specify explicitly the attributes that you want the cmdlet to set in the directory. Supply a list of the attribute LDAP display names as the parameter value. When used together with UseDefaultExcludedProperties, this parameter allows you to have the cmdlet set some attributes that would not be set otherwise.

Note: If a particular attribute is listed in both ExcludedProperties and IncludedProperties, the cmdlet does not set the value of that attribute in the directory.

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet sets the specified attributes to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to set. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

PasswordNeverExpires

Set this parameter to 'true' to configure the account so that its password never expires.

TsAllowLogon

Specify whether the user is allowed to log on to the Terminal Services. Parameter value can be 'true' or 'false':

- 'true' if logon is allowed
- 'false' if logon is not allowed

TsBrokenConnectionAction

Specify the action to take when a Terminal Services session limit is reached. Parameter value can be one of these integers:

- **1** (The client session should be terminated.)
- **0** (The client session should be disconnected.)

TsConnectClientDrives

Specify whether to reconnect to mapped client drives at logon to the Terminal Services. Parameter value can be 'true' or 'false':

- 'true' if reconnection is enabled
- 'false' if reconnection is disabled

TsConnectPrinterDrives

Specify whether to reconnect to mapped client printers at logon to the Terminal Services. Parameter value can be 'true' or 'false':

- 'true' if reconnection is enabled
- 'false' if reconnection is disabled

TsDefaultToMainPrinter

Specify whether to print automatically to the client's default printer. Parameter value can be 'true' or 'false':

- 'true' if printing to the client's default printer is enabled
- 'false' if printing to the client's default printer is disabled

TsHomeDirectory

Set the path to the Terminal Services home directory for the user. To set a home directory on the local computer, specify a local path; for example, C:\Path. To set a home directory in a network environment, set the TsHomeDrive parameter and specify a UNC path.

TsHomeDrive

Set the Terminal Services home drive for the user in a network environment. Parameter value is a string containing a drive letter followed by a colon, to which the UNC path for the Terminal Services home directory is mapped. To set a home directory in a network environment, set both this parameter and the TsHomeDirectory parameter.

TsInitialProgram

Set the path and file name of the application that starts automatically when the user logs on to the Terminal Services. To set an initial application to start when the user logs on, set both this parameter and the TsWorkDirectory parameter.

TsMaxConnectionTime

Set maximum duration of the Terminal Services session. After the specified time span has elapsed, the session can be disconnected or terminated. Parameter value is a TimeSpan object that specifies the duration you want.

TsMaxDisconnectionTime

Set maximum amount of time that a disconnected Terminal Services session remains active on the server. After the specified time span has elapsed, the session is terminated. Parameter value is a TimeSpan object that specifies the amount of time you want.

TsMaxIdleTime

Set maximum amount of time that the Terminal Services session can remain idle. After the specified time span has elapsed, the session can be disconnected or terminated. Parameter value is a TimeSpan object that specifies the amount of time you want.

TsProfilePath

Set a roaming or mandatory profile path to use when the user logs on to the Terminal Services. A valid parameter value is a string in the following network path format:
\\ServerName\ProfilesFolderName\UserName

TsReconnectionAction

Specify whether to allow reconnection to a disconnected Terminal Services session from any client computer. Parameter value can be one of these integers:

- **1** (Reconnection is allowed from the original client computer only.)
- **0** (Reconnection from any client computer is allowed.)

TsRemoteControl

Specify whether to allow remote observation or remote control of the user's Terminal Services session. Parameter value can be one of these integers:

- **0** (Remote control is disabled.)
- **1** (The user of remote control has full control of the user's session, with the user's permission.)
- **2** (The user of remote control has full control of the user's session; the user's permission is not required.)
- **3** (The user of remote control can view the session remotely, with the user's permission; the remote user cannot actively control the session.)
- **4** (The user of remote control can view the session remotely, but not actively control the session; the user's permission is not required.)

TsWorkDirectory

Set the Terminal Services working directory path for the user. To set an initial application to start when the user logs on to the Terminal Services, set both this parameter and the TsInitialProgram parameter.

UseDefaultExcludedProperties

When set to 'true', this parameter causes the cmdlet not to make changes to certain attributes in the directory. This pre-defined set of attributes (referred to as "default excluded properties") can be viewed or modified by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

UseDefaultExcludedPropertiesExcept

This parameter is deprecated, and has no effect.

UserMustChangePassword

Set this parameter to 'true' to configure the user account so that the user is required to change the password upon the next logon.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Attribute-specific Parameters

This cmdlet takes a series of optional, attribute-specific parameters allowing you to make changes to user attributes in Active Directory. Thus, to modify the value of the "givenName", "sn", or "l" attribute, you can use the *FirstName*, *LastName*, or *City* parameter, respectively.

If a particular attribute is referred to by both the ObjectAttributes array and the attribute-specific parameter, the ObjectAttributes setting has no effect on that attribute. The cmdlet sets the attribute to the value specified by the attribute-specific parameter.

The following table lists the attribute-specific parameters you can use with this cmdlet to manage user attributes. Each parameter is intended to manage a certain attribute, identified by the LDAP display name in the table. By using the cmdlet, you can set the value of the attribute to the parameter value specified. To clear the attribute, specify '' (empty string) as the parameter value.

TO MANAGE THIS ATTRIBUTE...	USE THIS SYNTAX
l	-City <String>
company	-Company <String>
description	-Description <String>
department	-Department <String>
displayName	-DisplayName <String>
facsimileTelephoneNumber	-Fax <String>
givenName	-FirstName <String>
homeDirectory	-HomeDirectory <String>
homeDrive	-HomeDrive <String>
homePhone	-HomePhone <String>
initials	-Initials <String>
sn	-LastName <String>
mail	-Email <String>
manager	-Manager <IdentityParameter>
mobile	-MobilePhone <String>
info	-Notes <String>
pager	-Pager <String>
physicalDeliveryOfficeName	-Office <String>
profilePath	-ProfilePath <String>

Use this parameter to set user password	-UserPassword <String>
scriptPath	-LogonScript <String>
telephoneNumber	-Phone <String>
postalCode	-PostalCode <String>
postOfficeBox	-PostOfficeBox <String>
samAccountName	-SamAccountName <String>
st	-StateOrProvince <String>
streetAddress	-StreetAddress <String>
title	-Title <String>
userPrincipalName	-UserPrincipalName <String>
wWWHomePage	-WebPage <String>

Detailed Description

Use this cmdlet to change or remove values of attributes of a user account in Active Directory.

Examples

Example 1

Connect to any available domain controller with the credentials of the locally logged on user, bind to a specific user account by DN, and modify the user description:

```
C:\PS> set-QADUser 'CN=John Smith,OU=CompanyOU,DC=company,DC=com' -description 'Sales
person'
```

Example 2

Connect to a specific domain controller with the credentials of a specific user, bind to a certain user account by SID, modify the user description, and then disconnect:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-QADService -service 'server.company.com' -ConnectionAccount
'company\administrator' -ConnectionPassword $pw
C:\PS> set-QADUser 'S-1-5-21-1279736177-1630491018-182859109-1305' -description
'Service account'
C:\PS> disconnect-QADService
```

Example 3

Connect to the local Administration Service with the credentials of a specific user, bind to a certain user account by Domain\Name, set or clear certain attributes, and then disconnect:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-QADService -service 'localhost' -proxy -ConnectionAccount
'company\administrator' -ConnectionPassword $pw
C:\PS> set-QADUser -identity 'company\jsmith' -ObjectAttributes @{l='New
York';description=''} -UserPassword 'P@ssword'
C:\PS> disconnect-QADService
```

Example 4

Assign two values to a multi-valued attribute such as "otherTelephone". This replaces the current values of the attribute with the specified values:

```
C:\PS> Set-QADUser 'mycompany.com/usersOU/User1' -objectAttributes
@{otherTelephone=@('555-34-67','555-34-68')}
```

Example 5

Add two values to a multi-valued attribute such as "otherTelephone". This appends the specified values to the existing values of the attribute. The existing values are not removed:

```
C:\PS> [Collections.DictionaryEntry] $de = new-object Collections.DictionaryEntry
-argumentList Append, @('555-34-67','555-34-68')
C:\PS> Set-QADUser 'mycompany.com/usersOU/User1' -objectAttributes
@{otherTelephone=$de}
```

Example 6

Delete the specified values from a multi-valued attribute such as "otherTelephone", leaving the other attribute values intact:

```
C:\PS> [Collections.DictionaryEntry] $de = new-object Collections.DictionaryEntry
-argumentList Delete, @('555-34-67','555-34-68')
C:\PS> Set-QADUser 'mycompany.com/usersOU/User1' -objectAttributes
@{otherTelephone=$de}
```

Example 7

Delete all values from a multi-valued attribute such as "otherTelephone" (clear the attribute on the user object):

```
C:\PS> Set-QADUser 'mycompany.com/usersOU/User1' -objectAttributes @{otherTelephone=''}
```

Example 8

Connect to the AD LDS instance on 'server.domain.local:389' with the credentials of the locally logged on user, bind to a specific AD LDS user object by DN, and modify the description of the AD LDS user object:

```
C:\PS> set-QADUser '<DN of user object>' -Service 'server.domain.local:389' -description
'My AD LDS user object'
```

New-QADUser

Create a new user account in Active Directory. Supported are both Active Directory Domain Services (AD DS) and Active Directory Lightweight Directory Services (AD LDS).

Syntax

```
New-QADUser [-Name] <String> -ParentContainer <IdentityParameter> [-City <String>]
[-Company <String>] [-Department <String>] [-Fax <String>] [-FirstName <String>]
[-HomePhone <String>] [-Initials <String>] [-LastName <String>] [-Manager
<IdentityParameter>] [-MobilePhone <String>] [-Notes <String>] [-Office <String>]
[-Pager <String>] [-PhoneNumber <String>] [-PostalCode <String>] [-PostOfficeBox
<String>] [-SamAccountName <String>] [-StateOrProvince <String>] [-StreetAddress
<String>] [-Title <String>] [-UserPrincipalName <String>] [-WebPage <String>]
[-UserPassword <String>] [-ObjectAttributes <ObjectAttributesParameter>] [-Description
<String>] [-DisplayName <String>] [-ExcludedProperties <String[]>] [-IncludedProperties
<String[]>] [-DeserializeValues] [-UseDefaultExcludedProperties]
[-UseDefaultExcludedPropertiesExcept <String[]>] [-Proxy] [-UseGlobalCatalog] [-Service
<String>] [-ConnectionAccount <String>] [-ConnectionPassword <SecureString>]
[-Credential <PSCredential>] [-Connection <ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has a number of optional parameters for managing individual attributes in the directory, with each parameter name identifying a certain attribute that can be set to a value specified by using the respective parameter (see the list of parameters for this cmdlet).

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Confirm

Prompts you for confirmation before executing the command.

DeserializeValues

Supply this parameter if the input you pass to the cmdlet contains serialized attribute values (for instance, when importing a directory object from a text file that was created using the Serialize parameter). For examples of how to export and import an object, see documentation on the Get-QADUser cmdlet.

ExcludedProperties

Use this parameter to specify the attributes that you do not want the cmdlet to set in the directory. Supply a list of the attribute LDAP display names as the parameter value. You could use this parameter when importing attribute values from a text file, in order to prevent some attributes found in the file from being set in the directory.

IncludedProperties

Use this parameter to specify explicitly the attributes that you want the cmdlet to set in the directory. Supply a list of the attribute LDAP display names as the parameter value. When used together with UseDefaultExcludedProperties, this parameter allows you to have the cmdlet set some attributes that would not be set otherwise.

Note: If a particular attribute is listed in both ExcludedProperties and IncludedProperties, the cmdlet does not set the value of that attribute in the directory.

Name

Set the 'name' attribute to this parameter value on the new object created by this cmdlet in the directory.

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet sets the specified attributes to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to set. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

ParentContainer

Specify the Distinguished Name of the container in which you want the new directory object to be created by this cmdlet.

UseDefaultExcludedProperties

When set to 'true', this parameter causes the cmdlet not to make changes to certain attributes in the directory. This pre-defined set of attributes (referred to as "default excluded properties") can be viewed or modified by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

UseDefaultExcludedPropertiesExcept

This parameter is deprecated, and has no effect.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Attribute-specific Parameters

This cmdlet takes a series of optional, attribute-specific parameters allowing you to set attributes in the newly created account. Thus, to set the value of the "givenName", "sn", or "l" attribute, you can use the *FirstName*, *LastName*, or *City* parameter, respectively.

If a particular attribute is referred to by both the ObjectAttributes array and the attribute-specific parameter, the ObjectAttributes setting has no effect on that attribute. The cmdlet sets the attribute to the value specified by the attribute-specific parameter.

The following table lists the attribute-specific parameters you can use with this cmdlet to manage user attributes. Each parameter is intended to manage a certain attribute, identified by the LDAP display name in the table. By using the cmdlet, you can set the value of the attribute to the parameter value specified.

TO MANAGE THIS ATTRIBUTE...	USE THIS SYNTAX
l	-City <String>
company	-Company <String>
description	-Description <String>
department	-Department <String>
displayName	-DisplayName <String>
facsimileTelephoneNumber	-Fax <String>
givenName	-FirstName <String>
homePhone	-HomePhone <String>
initials	-Initials <String>
sn	-LastName <String>
manager	-Manager <IdentityParameter>
mobile	-MobilePhone <String>
info	-Notes <String>
physicalDeliveryOfficeName	-Office <String>
pager	-Pager <String>
Use this parameter to set user password	-UserPassword <String>
telephoneNumber	-Phone <String>
postalCode	-PostalCode <String>
postOfficeBox	-PostOfficeBox <String>
samAccountName	-SamAccountName <String>
st	-StateOrProvince <String>
streetAddress	-StreetAddress <String>
title	-Title <String>
userPrincipalName	-UserPrincipalName <String>
wWWHomePage	-WebPage <String>

Detailed Description

Use this cmdlet to create a user account in Active Directory and, optionally, set attribute values in the newly created account.

Examples

Example 1

Connect to any available domain controller with the credentials of the locally logged on user, create a new user account, and set a password for the new account:

```
C:\PS> new-QADUser -name 'user1' -ParentContainer 'OU=companyOU,DC=company,DC=com'
-SamAccountName 'user1' -UserPassword 'P@ssword'
```

Example 2

Connect to the local Administration Service with the credentials of a specific user, create a new user account, set a password for the new account, and then disconnect (this example assumes that a value for the sAMAccountName attribute is to be generated by ActiveRoles Server, based on a provisioning policy):

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-QADService -service 'localhost' -proxy -ConnectionAccount
'company\administrator' -ConnectionPassword $pw
C:\PS> new-QADUser -name 'user1' -ParentContainer 'OU=companyOU,DC=company,DC=com'
-UserPassword 'P@ssword'
C:\PS> disconnect-QADService
```

Example 3

Connect to the local Administration Service with the credentials of a specific user, import a CSV file, for each record in the file create a new user account with the name matching the value in the 'user name' column in the CSV file, and then disconnect:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-qadService -service 'localhost' -proxy -ConnectionAccount
'company\administrator' -ConnectionPassword $pw
C:\PS> import-csv C:\temp\data.csv | %{new-qadUser -ParentContainer
'OU=companyOU,DC=company,DC=com' -name $_.'user name'}
C:\PS> disconnect-qadService
```

In this example, the **%** character preceding the script block is an alias for the **ForEach-Object** cmdlet. The sAMAccountName attribute is assumed to be set by ActiveRoles Server. For more information on this example, refer to the “Pipelining” section earlier in this document.

Example 4

Connect to the AD LDS instance on 'server.domain.local:389' with the credentials of the locally logged on user, and create a new AD LDS user object in a certain container:

```
C:\PS> new-QADUser -Service 'server.domain.local:389' -Name 'user1' -ParentContainer
'<DN of container>' -UserPassword 'P@ssword'
```

Disable-QADUser

Disable a user account in Active Directory. Supported are both Active Directory Domain Services (AD DS) and Active Directory Lightweight Directory Services (AD LDS).

Syntax

```
Disable-QADUser [[-Identity] <IdentityParameter>] [-Proxy] [-UseGlobalCatalog] [-Service
<String>] [-ConnectionAccount <String>] [-ConnectionPassword <SecureString>]
[-Credential <PSCredential>] [-Connection <ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Confirm

Prompts you for confirmation before executing the command.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to disable a user account in Active Directory Domain Services or Active Directory Lightweight Directory Services.

Examples

Example 1

Connect to any available domain controller with the credentials of the locally logged on user and disable the user account identified by Domain\Name:

```
C:\PS> disable-QADUser 'MyDomain\JSmith'
```

Example 2

Connect to the AD LDS instance on 'server.domain.local:389' with the credentials of the locally logged on user, and disable the AD LDS user account that is identified by DN:

```
C:\PS> disable-QADUser '<DN of user account>' -Service 'server.domain.local:389'
```

Enable-QADUser

Enable a user account in Active Directory. Supported are both Active Directory Domain Services (AD DS) and Active Directory Lightweight Directory Services (AD LDS).

Syntax

```
Enable-QADUser [[-Identity] <IdentityParameter>] [-Proxy] [-UseGlobalCatalog] [-Service
<String>] [-ConnectionAccount <String>] [-ConnectionPassword <SecureString>]
[-Credential <PSCredential>] [-Connection <ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Confirm

Prompts you for confirmation before executing the command.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to re-enable a disabled user account in Active Directory Domain Services or Active Directory Lightweight Directory Services.

Examples

Example 1

Connect to any available domain controller with the credentials of the locally logged on user and enable the user account identified by Domain\Name:

```
C:\PS> enable-QADUser 'MyDomain\JSmith'
```

Example 2

Connect to the AD LDS instance on 'server.domain.local:389' with the credentials of the locally logged on user, and enable the AD LDS user account that is identified by DN:

```
C:\PS> enable-QADUser '<DN of user account>' -Service 'server.domain.local:389'
```

Unlock-QADUser

Unlock a user account in Active Directory. Supported are both Active Directory Domain Services (AD DS) and Active Directory Lightweight Directory Services (AD LDS).

Syntax

```
Unlock-QADUser [[-Identity] <IdentityParameter>] [-Proxy] [-UseGlobalCatalog] [-Service
<String>] [-ConnectionAccount <String>] [-ConnectionPassword <SecureString>]
[-Credential <PSCredential>] [-Connection <ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Confirm

Prompts you for confirmation before executing the command.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to unlock a user account that has been locked out due to a number of failed logon attempts. You can unlock user accounts in both Active Directory Domain Services (AD DS) and Active Directory Lightweight Directory Services (AD LDS).

Examples

Example 1

Connect to any available domain controller with the credentials of the locally logged on user and unlock the user account identified by Domain\Name:

```
C:\PS> unlock-QADUser 'MyDomain\JSmith'
```

Example 2

Connect to the AD LDS instance on 'server.domain.local:389' with the credentials of the locally logged on user, and unlock the AD LDS user account that is identified by DN:

```
C:\PS> unlock-QADUser '<DN of user account>' -Service 'server.domain.local:389'
```

Deprovision-QADUser

Request ActiveRoles Server to deprovision a user account. This cmdlet requires a connection to be established to the ActiveRoles Server Administration Service by supplying the *Proxy* parameter.

Syntax

```
Deprovision-QADUser [-Identity] <IdentityParameter> [-ReportFile <String>] [-Xml]
[-Proxy] [-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Note that this cmdlet requires a connection to the ActiveRoles Server Administration Service, so the *Proxy* parameter must be used to establish a connection.

Parameters

Confirm

Prompts you for confirmation before executing the command.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

ReportFile

Supply this parameter if you want to save a report on the deprovisioning results to a file in HTML or XML format. The parameter value must be a valid path to a file, including the file name. The cmdlet creates the file if necessary. Omit this parameter if you do not want to save the report in a file. ActiveRoles Server preserves the report data regardless of this parameter, so you always have the option to examine the deprovisioning results using the ActiveRoles Server console.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

XML

Supply this parameter if you want to save the report on the deprovisioning results in XML format. Omit this parameter if you want to save the report in HTML format.

Detailed Description

Use this cmdlet to deprovision a user account via ActiveRoles Server. ActiveRoles Server provides the ability to deprovision rather than delete or only disable user accounts. Deprovision refers to a set of actions being performed in order to revoke user access to resources. The deprovision operation on user objects triggers deprovision policies. ActiveRoles Server comes with a default policy to automate some commonly-used deprovisioning tasks, and allows the deprovision policies to be adjusted as needed.

This cmdlet originates a request to deprovision the user accounts specified. When processing the request, ActiveRoles Server performs all operations prescribed by the deprovision policies.

Examples

Example 1

Connect to any available ActiveRoles Server Administration Service with the credentials of the locally logged on user and deprovision the user account identified by Domain\Name:

```
C:\PS> deprovision-QADUser 'MyDomain\JSmith' -Proxy
```

Example 2

Connect to a specific Administration Service with the credentials of the locally logged on user, retrieve a user object using the get-QADUser cmdlet, and pipe the user object into the Deprovision-QADUser cmdlet to deprovision the user account represented by that object:

```
C:\PS> connect-QADService -Service 'myserver.mydomain.lab' -Proxy
C:\PS> get-QADUser 'MyDomain\JSmith' | deprovision-QADUser
```

Example 3

Connect to a specific Administration Service with the credentials of the locally logged on user, retrieve a user object using the get-QADUser cmdlet, and pipe the user object into the Deprovision-QADUser cmdlet to deprovision the user account represented by that object, with a report on the deprovisioning results being saved in a specific file in HTML format:

```
C:\PS> connect-QADService -Service 'myserver.mydomain.lab' -Proxy
C:\PS> get-QADUser 'MyDomain\JSmith' | deprovision-QADUser -ReportFile 'C:\JSmith.html'
```

Example 4

Connect to a specific Administration Service with the credentials of the locally logged on user, and deprovision all user accounts found in a specific container, with a report on the deprovisioning results for each user account being saved in a separate file:

```
C:\PS> connect-QADService -Service 'myserver.mydomain.lab' -Proxy
C:\PS> get-QADUser -SearchRoot 'mydomain.lab/retired' | deprovision-QADUser -ReportFile
{'C:\DeprovisionReports\' + $_.SamAccountName + '.html'}
```

Get-QADGroup

Retrieve all groups in a domain or container that match the specified conditions. Supported are both Active Directory Domain Services (AD DS) and Active Directory Lightweight Directory Services (AD LDS).

Syntax

```
Get-QADGroup [[-Identity] <IdentityParameter>] [-SamAccountName <String>] [-GroupType
<GroupType>] [-GroupScope <GroupScope>] [-Dynamic] [-Empty] [-ContainsMember
<IdentityParameter[]>] [-ContainsIndirectMember <IdentityParameter[]>]
[-NotContainsMember <IdentityParameter[]>] [-NotContainsIndirectMember
<IdentityParameter[]>] [-MemberOf <IdentityParameter[]>] [-IndirectMemberOf
<IdentityParameter[]>] [-NotMemberOf <IdentityParameter[]>] [-NotIndirectMemberOf
<IdentityParameter[]>] [-Tombstone] [-LastKnownParent <IdentityParameter>]
[-SecurityMask <SecurityMasks>] [-SearchRoot <IdentityParameter>] [-SearchScope
<SearchScope>] [-AttributeScopeQuery <String>] [-PageSize <Int32>] [-SizeLimit <Int32>]
[-LdapFilter <String>] [-WildcardMode <WildcardMode>] [-ObjectAttributes <Object>]
[-Description <String>] [-DisplayName <String>] [-Name <String>] [-Anr <String>]
[-CreatedOn <DateTime>] [-CreatedAfter <DateTime>] [-CreatedBefore <DateTime>]
[-LastChangedOn <DateTime>] [-LastChangedAfter <DateTime>] [-LastChangedBefore
<DateTime>] [-IncludeAllProperties] [-DontConvertValuesToFriendlyRepresentation]
[-SerializeValues] [-ReturnPropertyNamesOnly] [-DontUseDefaultIncludedProperties]
[-UseDefaultExcludedProperties] [-ExcludedProperties <String[]>] [-IncludedProperties
<String[]>] [-UseDefaultExcludedPropertiesExcept <String[]>] [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Anr

Specify a value to be resolved using ambiguous name resolution (ANR). Which attributes are included in an ANR search depends upon the Active Directory schema. Thus, in Windows Server 2003 based Active Directory, the following attributes are set for ANR by default:

- Display-Name (displayName)
- Given-Name (givenName)
- Legacy-Exchange-DN (legacyExchangeDN)
- ms-DS-Additional-Sam-Account-Name (msDS-AdditionalSamAccountName)
- Physical-Delivery-Office-Name (physicalDeliveryOfficeName)
- Proxy-Addresses (proxyAddresses)

- RDN (name)
- SAM-Account-Name (sAMAccountName)
- Surname (sn)

For instance, when you supply 'ann*' as the value of this parameter, the cmdlet searches for objects that have ann at the beginning of the value of at least one of the attributes listed above.

AttributeScopeQuery

Specify the LDAP display name of an attribute that has DN syntax (for example, "member" or "memberOf"). The cmdlet enumerates the Distinguished Name values of the attribute on the object specified by the SearchRoot parameter, and performs the search on the objects represented by the Distinguished Names. The SearchScope parameter has no effect in this case. The object to search must be specified by using the SearchRoot parameter rather than the Identity parameter.

For instance, with the value of this parameter set to "memberOf", the cmdlet searches the collection of the groups to which the SearchRoot object belongs.

ContainsIndirectMember

Retrieve groups to which the object or objects specified by this parameter belong, whether directly or because of group nesting. The cmdlet returns a group if the object has direct or indirect membership in that group.

ContainsMember

Retrieve groups that hold the object or objects specified by this parameter. The cmdlet returns a group if the object is a direct member of that group.

CreatedAfter

Specify the lower boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created after the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedBefore

Specify the upper boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created before the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedOn

Specify the object creation date by which to filter objects found, searching for objects created within the date specified. This parameter is mutually exclusive with the CreatedAfter and CreatedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

Description

Search by the 'description' attribute.

DisplayName

Search by the 'displayName' attribute.

DontConvertValuesToFriendlyRepresentation

This parameter causes the cmdlet to represent the Integer8 and OctetString attribute values "as is," without converting them to a user-friendly, human-readable form. If this parameter is omitted, the cmdlet performs the following data conversions:

- The values of the Integer8 attributes listed in the Integer8AttributesThatContainDateTimes array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to DateTime.
- The values of the Integer8 attributes listed in the Integer8AttributesThatContainNegativeTimeSpans array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to TimeSpan.
- The values of the other Integer8 attributes are converted from IADsLargeInteger to Int64.
- The values of the OctetString attributes are converted from byte[] to BinHex strings.

Note: This parameter has an effect only on the properties of the output object that have the member type of NoteProperty. Such properties are normally added to the output object in order to provide access to the attribute values of the respective directory object that are loaded to the local memory cache but cannot be accessed by using properties of the base object (the object for which the output object serves as a wrapper).

DontUseDefaultIncludedProperties

This parameter causes the cmdlet to load only a small set of attributes from the directory to the local memory cache (normally, this set is limited to objectClass and ADsPath). Other attributes are retrieved from the directory as needed when you use the cmdlet output objects to read attribute values. Thus, if you want only to count the objects that meet certain conditions (rather than examine values of particular attributes), then you can use this parameter to increase performance of your search. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

Empty

Set this parameter to 'true' for the cmdlet to retrieve only those groups that have no members (empty groups).

Note: A group is considered empty if it has the "member" attribute not set. So, the Empty parameter can retrieve a group that has only those members for which the group is set as the primary group. An example is the Domain Users group, which normally is the primary group for any user account while having the "member" attribute not set.

ExcludedProperties

Use this parameter to specify the attributes that you do not want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the ExcludedProperties parameter you can change this default behavior on an ad-hoc basis, in order to prevent certain attributes from being loaded. Another scenario involves the use of this parameter in conjunction with IncludeAllProperties in order to restrict the set of the cached attributes.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

GroupScope

Specify the group scope of groups to find. Acceptable values are:

- 'Global'
- 'Universal'
- 'DomainLocal'

GroupType

Specify the group type of groups to find. Acceptable values are:

- 'Security'
- 'Distribution'

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to find.

The cmdlet attempts to retrieve the object identified by this parameter value, disregarding the other search parameters. If you want other search parameters to have effect, do not supply this parameter.

IncludeAllProperties

With this parameter, the cmdlet retrieves all attributes of the respective directory object (such as a User object), and stores the attribute values in the memory cache on the local computer. Attribute values can be read from the memory cache by using properties of the object returned by the cmdlet. Thus, when used in conjunction with the SerializeValues parameter, it allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

IncludedProperties

Use this parameter to specify the attributes that you want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the IncludedProperty parameter you can direct the cmdlet to cache some attributes in addition to the default set.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet.

IndirectMemberOf

Retrieve objects that belong to the group or groups specified by this parameter, whether directly or because of group nesting. The cmdlet returns an object if the object has direct or indirect membership in the group specified by this parameter value.

LastChangedAfter

Specify the lower boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed after the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedBefore

Specify the upper boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed before the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedOn

Specify the object modification date by which to filter objects found, searching for objects that have last changed within the date specified. This parameter is mutually exclusive with the LastChangedAfter and LastChangedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

LastKnownParent

When searching for a deleted object by using the Tombstone parameter, specify the DN of the container the object was in before it became a tombstone. This allows you to find objects that were deleted from a particular container.

Note that the lastKnownParent attribute is only set if the object was deleted on a domain controller running Windows Server 2003 or later version of Microsoft Windows Server. Therefore, it is possible that the lastKnownParent attribute value is inaccurate.

LdapFilter

Specify the LDAP search filter that defines your search criteria. Note that the search filter string is case-sensitive.

The cmdlet disregards this parameter if an Identity value is supplied. If you want this parameter to have effect, do not supply an Identity parameter value.

If you supply the LdapFilter parameter along with attribute-specific parameters, then your search returns objects that meet the conditions defined by the LDAP filter and have the specified attributes set to the specified values.

MemberOf

Retrieve objects that are direct members of the group or groups specified by this parameter. The cmdlet returns an object if the object has direct membership in the group specified by this parameter value.

Name

Search by the 'name' attribute.

NotContainsIndirectMember

Retrieve groups to which the object or objects specified by this parameter do not belong directly or because of group nesting. The cmdlet returns a group if the object has neither direct nor indirect membership in that group.

NotContainsMember

Retrieve groups that do not hold the object or objects specified by this parameter. The cmdlet returns a group if the object is not a direct member of that group.

NotIndirectMemberOf

Retrieve objects that do not belong to the group or groups specified by this parameter, whether directly or because of group nesting. The cmdlet returns an object if the object has neither direct nor indirect membership in the group specified by this parameter value.

NotMemberOf

Retrieve objects that are not direct members of the group or groups specified by this parameter. The cmdlet returns an object if the object does not have direct membership in the group specified by this parameter value.

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet searches for objects that have the specified attributes set to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to search. A value may include an asterisk character - a wildcard representing any group of characters. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

PageSize

Set the maximum number of items in each page of the search results that will be returned by the cmdlet. After the directory server has found the number of objects that are specified by this parameter, it will stop searching and return the results to the cmdlet. When the cmdlet requests more data, the server will restart the search where it left off. You can use this setting to adjust the number of requests (network calls) to the directory server issued by the cmdlet during a search.

Normally, the default page size is 50. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

ReturnPropertyNamesOnly

This parameter causes the cmdlet to list the names of the object attributes whose values the cmdlet retrieves from the directory and stores in the memory cache on the local computer. Thus, when used in conjunction with the IncludeAllProperties parameter, it lists the names of all attributes of the respective directory object (such as a User object). For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

SamAccountName

Search by the 'sAMAccountName' attribute (pre-Windows 2000 name).

SearchRoot

Specify the Distinguished Name, Canonical Name, or GUID of the container to search. By default, the cmdlet searches the entire sub-tree of which SearchRoot is the topmost object (sub-tree search). This default behavior can be altered by using the SearchScope parameter.

The search criteria are defined by the LdapFilter parameter value and the values of attribute-specific parameters.

The cmdlet disregards this parameter if an Identity value is supplied. If you want this parameter to have effect, do not supply any Identity parameter value.

SearchScope

Specify one of these parameter values:

- 'Base' - Limits the search to the base (SearchRoot) object. The result contains a maximum of one object.
- 'OneLevel' - Searches the immediate descendant (child) objects of the base (SearchRoot) object, excluding the base object.
- 'Subtree' - Searches the whole sub-tree, including the base (SearchRoot) object and all its descendant objects.

Normally, if this parameter is not supplied, the cmdlet performs a Subtree search. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

SecurityMask

Specify which elements of the object’s security descriptor to retrieve. Valid parameter values are:

- 'None' - do not retrieve any security data
- 'Owner' - retrieve the owner data
- 'Group' - retrieve the primary group data
- 'Dacl' - retrieve the discretionary access-control list data
- 'Sacl' - retrieve the system access-control list data

You can supply a combination of these values, separating them by commas. For example, you can supply the parameter value of 'Dacl,Sacl' in order to retrieve both the discretionary and system access-control list data.

SerializeValues

This parameter causes the cmdlet to output an object whose properties store the attribute values of the respective directory object that are loaded to the local memory cache. The value returned by each property of the output object is represented as a string (serialized) so as to facilitate the export of the attribute values to a text file. Thus, when used in conjunction with the IncludeAllProperties parameter, this parameter allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

SizeLimit

Set the maximum number of items to be returned by the cmdlet. Normally, the default size limit is 1000. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

Tombstone

Search for deleted objects of the respective object class. The search output is normally intended to be passed (piped in) to the Restore-QADDeletedObject cmdlet for restoring deleted objects.

When Active Directory deletes an object from the directory, the object becomes a tombstone. A search using this parameter returns tombstone objects that meet the filtering criteria supplied. Note that upon deletion of an object only a small number of the object's attributes are saved in the tombstone, with most of the attributes being lost. To search for deleted objects, your search filter should be based on the attributes that are preserved in tombstones. Which attributes are preserved is determined by Active Directory. Thus, the attributes that have the 0x8 bit set in the searchFlags attribute of the attributeSchema definition are preserved.

When the Tombstone parameter is supplied, the search results include deleted objects (tombstones) that match the specified search filter. However, a search filter that matches a live object may not work as expected after the object is deleted. This is because not all attributes are retained in the tombstone. For example, a filter such as (&(objectClass=user)(objectCategory=person)) would not match any tombstone objects since the objectCategory attribute is removed upon object deletion. Conversely, the objectClass attribute is retained on tombstone objects, so a filter of (objectClass=user) would match deleted user objects.

The name (RDN) of a tombstone object begins with the name (RDN) of the deleted object, so a search using the Tombstone parameter can be refined by adding a filter based on object name. For example, to search for deleted objects with a name that begins with "John", you can use a filter such as (cn=John*).

It is also possible to find a specific deleted object. If you know the name (RDN) of the object and the Distinguished Name (DN) of the container the object was in before it was deleted, then you can pass the container's DN to the LastKnownParent parameter and apply a filter of (cn=<RDN of the object>*) in order to have the cmdlet retrieve that specific object. However, if an object is deleted, a new object with the same DN is created, and then deleted as well, the above search would return more than one tombstone object. The returned tombstone objects are distinguished by the GUIDs of the deleted objects, with the name of each tombstone object ending in the GUID of the respective deleted object.

UseDefaultExcludedProperties

When set to 'true', this parameter causes the cmdlet not to load a certain pre-defined set of attributes from the directory to the local memory cache. This pre-defined set of attributes (referred to as "default excluded properties") can be viewed or modified by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Normally, this parameter is used in conjunction with IncudeAllProperties to avoid retrieval of unnecessary data from the directory server, thereby increasing performance of the search operation performed by the cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

UseDefaultExcludedPropertiesExcept

This parameter is deprecated, and has no effect.

WildcardMode

Specify either 'PowerShell' or 'LDAP' as the parameter value. Normally, if this parameter is not supplied, the cmdlet assumes that WildcardMode is set to 'LDAP'. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. The 'PowerShell' value causes the cmdlet to use PowerShell wildcards and quoting rules. Wildcards are processed on the client side, which may result in slow search performance.

For information about PowerShell wildcards and quoting rules, type the following commands at the PowerShell command-prompt:

```
help about_wildcard
help about_quoting_rule
```

The 'LDAP' value causes the cmdlet to use LDAP wildcards (asterisks only) and LDAP quoting rules (backslash as the escape character). Wildcards are processed on the server side, which enables faster search results.

Detailed Description

Use this cmdlet to search an Active Directory domain or container for groups that meet certain criteria, or to bind to a certain group by DN, SID, GUID, or Domain\Name. You can search by group attributes or specify your search criteria by using an LDAP search filter.

The output of the cmdlet is a collection of objects, with each object representing one of the groups found by the cmdlet. You can pipe the output into another cmdlet, such as **Set-QADObject**, to make changes to the groups returned by this cmdlet.

Examples

Example 1

Connect to any available domain controller with the credentials of the locally logged on user, bind to a specific group by Domain\Name, and display the description of the group:

```
C:\PS> (get-QADGroup 'MyDom\Administrators').DirectoryEntry.description
```

Example 2

Connect to a specific domain controller with the credentials of a specific user, bind to a certain group by SID, display the description of the group, and then disconnect:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-QADService -service 'server.company.com' -ConnectionAccount
'company\administrator' -ConnectionPassword $pw
C:\PS> (get-QADGroup -identity
'S-1-5-21-1279736177-1630491018-182859109-1305').DirectoryEntry.description
C:\PS> disconnect-QADService
```

Example 3

Connect to any available domain controller with the credentials of the locally logged on user, search for groups in a specific container by using an LDAP search filter, and display a list of the groups found:

```
C:\PS> get-QADGroup -SearchRoot 'company.com/GroupsOU' -LdapFilter '(description=a*)'
```

Example 4

Connect to any available domain controller with the credentials of the locally logged on user, find all distribution groups in a specific container, and display a list of the groups found:

```
C:\PS> get-QADGroup -SearchRoot 'company.com/GroupsOU' -GroupType 'Distribution'
```

Example 5

Connect to any available domain controller with the credentials of a specific user, search a certain container to find all groups with the empty Notes field, set a note for each of those groups, and then disconnect:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-QADService -ConnectionAccount 'company\administrator'
-ConnectionPassword $pw
C:\PS> get-QADGroup -SearchRoot 'company.com/GroupsOU' -ObjectAttributes @{info=''} |
set-QADObject -ObjectAttributes @{info='A note'}
C:\PS> disconnect-QADService
```

Example 6

Connect to the AD LDS instance on 'server.domain.local:389' with the credentials of the locally logged on user, find all AD LDS groups in a specific container, and display a list of the groups found:

```
C:\PS> get-QADGroup -Service 'server.domain.local:389' -SearchRoot '<DN of container>'
```

Example 7

Retrieve groups from a particular container that have a particular user account as a direct or indirect member:

```
C:\PS> get-QADGroup -SearchRoot '<DN of container>' -ContainsIndirectMember
'domainName\userName'
```

Set-QADGroup

Modify attributes of a group in Active Directory. Supported are both Active Directory Domain Services (AD DS) and Active Directory Lightweight Directory Services (AD LDS).

Syntax

```
Set-QADGroup [-Identity] <IdentityParameter> [-ManagedBy <IdentityParameter>] [-Notes
<String>] [-Email <String>] [-GroupType <GroupType>] [-GroupScope <GroupScope>]
[-SamAccountName <String>] [-ObjectAttributes <ObjectAttributesParameter>]
[-Description <String>] [-DisplayName <String>] [-ExcludedProperties <String[]>]
[-IncludedProperties <String[]>] [-DeserializeValues] [-UseDefaultExcludedProperties]
[-UseDefaultExcludedPropertiesExcept <String[]>] [-Proxy] [-UseGlobalCatalog] [-Service
<String>] [-ConnectionAccount <String>] [-ConnectionPassword <SecureString>]
[-Credential <PSCredential>] [-Connection <ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Confirm

Prompts you for confirmation before executing the command.

Description

Set the 'description' attribute to this parameter value.

DeserializeValues

Supply this parameter if the input you pass to the cmdlet contains serialized attribute values (for instance, when importing a directory object from a text file that was created using the Serialize parameter). For examples of how to export and import an object, see documentation on the Get-QADUser cmdlet.

DisplayName

Set the 'displayName' attribute to this parameter value.

Email

Set the 'mail' attribute to this parameter value.

ExcludedProperties

Use this parameter to specify the attributes that you do not want the cmdlet to set in the directory. Supply a list of the attribute LDAP display names as the parameter value. You could use this parameter when importing attribute values from a text file, in order to prevent some attributes found in the file from being set in the directory.

GroupScope

Set the group scope to this parameter value. Acceptable values are:

- 'Global'
- 'Universal'
- 'DomainLocal'

GroupType

Set the group type to this parameter value. Acceptable values are:

- 'Security'
- 'Distribution'

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

IncludedProperties

Use this parameter to specify explicitly the attributes that you want the cmdlet to set in the directory. Supply a list of the attribute LDAP display names as the parameter value. When used together with UseDefaultExcludedProperties, this parameter allows you to have the cmdlet set some attributes that would not be set otherwise.

Note: If a particular attribute is listed in both ExcludedProperties and IncludedProperties, the cmdlet does not set the value of that attribute in the directory.

ManagedBy

Specify the DN, SID, GUID, UPN or Domain\Name of the user or group to be set as the 'managedBy' attribute value on the object in the directory.

Notes

Set the 'info' attribute to this parameter value.

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet sets the specified attributes to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to set. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

SamAccountName

Set the 'sAMAccountName' attribute (pre-Windows 2000 name) to this parameter value.

UseDefaultExcludedProperties

When set to 'true', this parameter causes the cmdlet not to make changes to certain attributes in the directory. This pre-defined set of attributes (referred to as "default excluded properties") can be viewed or modified by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

UseDefaultExcludedPropertiesExcept

This parameter is deprecated, and has no effect.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to change or remove values of attributes of a group in Active Directory.

The cmdlet takes a series of optional, attribute-specific parameters allowing you to make changes to attributes in Active Directory. Thus, to modify the value of the 'description' or 'displayName' attribute, you can use the *-Description* or *-DisplayName* parameter, respectively. If a particular attribute is referred to by both the ObjectAttributes array and an attribute-specific parameter, the ObjectAttributes setting has no effect on that attribute. The cmdlet sets the attribute to the value specified by the attribute-specific parameter.

Examples

Example 1

Connect to any available domain controller with the credentials of the locally logged on user, bind to a specific group by Domain\Name, and modify the description of the group:

```
C:\PS> set-QADGroup 'MyDomain\AMS Managers' -description 'Amsterdam Managers'
```

Example 2

Connect to the AD LDS instance on 'server.domain.local:389' with the credentials of the locally logged on user, bind to a specific AD LDS group object by DN, and modify the description of the AD LDS group object:

```
C:\PS> set-QADGroup '<DN of group object>' -Service 'server.domain.local:389'
-description 'My AD LDS group object'
```

Example 3

Pipe the get-QADGroup output into the set-QADGroup cmdlet to change the pre-Windows 2000 group name (add the "New" suffix to the name of the group returned by get-QADGroup):

```
C:\PS> Get-QADGroup MyTestGroup | %{Set-QADGroup $_ –SamAccountName ($_.SamAccountName
+ "New")}
```

Example 4

Bind to the group by distinguished name and set the group name (pre-Windows 2000):

```
C:\PS> set-QADGroup 'CN=TestGroup,OU=Groups,DC=domain,DC=company,DC=com'
-samaccountname 'My Test Group'
```

New-QADGroup

Create a new group in Active Directory. Supported are both Active Directory Domain Services (AD DS) and Active Directory Lightweight Directory Services (AD LDS).

Syntax

```
New-QADGroup [-Name] <String> -ParentContainer <IdentityParameter> [-Member
<IdentityParameter[]>] [-GroupType <GroupType>] [-GroupScope <GroupScope>]
[-SamAccountName <String>] [-ObjectAttributes <ObjectAttributesParameter>]
[-Description <String>] [-DisplayName <String>] [-ExcludedProperties <String[]>]
[-IncludedProperties <String[]>] [-DeserializeValues] [-UseDefaultExcludedProperties]
[-UseDefaultExcludedPropertiesExcept <String[]>] [-Proxy] [-UseGlobalCatalog] [-Service
<String>] [-ConnectionAccount <String>] [-ConnectionPassword <SecureString>]
[-Credential <PSCredential>] [-Connection <ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has a number of optional parameters for managing individual attributes in the directory, with each parameter name identifying a certain attribute that can be set to a value specified by using the respective parameter (see the list of parameters for this cmdlet).

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Confirm

Prompts you for confirmation before executing the command.

DeserializeValues

Supply this parameter if the input you pass to the cmdlet contains serialized attribute values (for instance, when importing a directory object from a text file that was created using the Serialize parameter). For examples of how to export and import an object, see documentation on the Get-QADUser cmdlet.

ExcludedProperties

Use this parameter to specify the attributes that you do not want the cmdlet to set in the directory. Supply a list of the attribute LDAP display names as the parameter value. You could use this parameter when importing attribute values from a text file, in order to prevent some attributes found in the file from being set in the directory.

GroupScope

Set the group scope to this parameter value. Acceptable values are:

- 'Global'
- 'Universal'
- 'DomainLocal'

GroupType

Set the group type to this parameter value. Acceptable values are:

- 'Security'
- 'Distribution'

IncludedProperties

Use this parameter to specify explicitly the attributes that you want the cmdlet to set in the directory. Supply a list of the attribute LDAP display names as the parameter value. When used together with UseDefaultExcludedProperties, this parameter allows you to have the cmdlet set some attributes that would not be set otherwise.

Note: If a particular attribute is listed in both ExcludedProperties and IncludedProperties, the cmdlet does not set the value of that attribute in the directory.

Name

Set the 'name' attribute to this parameter value on the new object created by this cmdlet in the directory.

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet sets the specified attributes to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to set. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

ParentContainer

Specify the Distinguished Name of the container in which you want the new directory object to be created by this cmdlet.

SamAccountName

Set the 'sAMAccountName' attribute (pre-Windows 2000 name) to this parameter value.

UseDefaultExcludedProperties

When set to 'true', this parameter causes the cmdlet not to make changes to certain attributes in the directory. This pre-defined set of attributes (referred to as "default excluded properties") can be viewed or modified by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

UseDefaultExcludedPropertiesExcept

This parameter is deprecated, and has no effect.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Attribute-specific Parameters

This cmdlet also takes a series of optional, attribute-specific parameters allowing you to set attributes in the newly created group. Thus, to set the value of the "description", "displayName", or "member" attribute, you can use the *Description*, *DisplayName*, or *Member* parameter, respectively.

If a particular attribute is referred to by both the ObjectAttributes array and the attribute-specific parameter, the ObjectAttributes setting has no effect on that attribute. The cmdlet sets the attribute to the value specified by the attribute-specific parameter.

The following table lists the attribute-specific parameters you can use with this cmdlet to manage attributes of group objects. Each parameter is intended to manage a certain attribute, identified by the LDAP display name in the table. By using the cmdlet, you can set the value of the attribute to the parameter value specified.

TO MANAGE THIS ATTRIBUTE...	USE THIS SYNTAX
description	-Description <String>
displayName	-DisplayName <String>
member	-Member <String[]>

Detailed Description

Use this cmdlet to create a group in Active Directory and, optionally, add members to and set other attribute values in the newly created group.

Examples

Example 1

Connect to any available domain controller with the credentials of the locally logged on user, and create a new universal distribution group:

```
C:\PS> new-qadGroup -ParentContainer 'OU=companyOU,DC=company,DC=com' -name 'group1'
-samAccountName 'group1' -grouptype 'Distribution' -groupscope 'Universal'
```

Example 2

Connect to the local Administration Service with the credentials of a specific user, create a new universal distribution group, and then disconnect:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-qadService -service 'localhost' -proxy -ConnectionAccount
'company\administrator' -ConnectionPassword $pw
C:\PS> new-qadGroup -ParentContainer 'OU=companyOU,DC=company,DC=com' -name 'group1'
-samAccountName 'group1' -grouptype 'Distribution' -groupscope 'Universal'
C:\PS> disconnect-qadService
```

Example 3

Connect to the local Administration Service with the credentials of a specific user, import a CSV file, for each record in the file create a new global security group with the name matching the value in the 'group name' column in the CSV file, and then disconnect:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-qadService -service 'localhost' -proxy -ConnectionAccount
'company\administrator' -ConnectionPassword $pw
C:\PS> import-csv C:\temp\data.csv | %{new-qadGroup -ParentContainer
'OU=companyOU,DC=company,DC=com' -name $_.'group name' -samAccountName $_.'group name'}
C:\PS> disconnect-qadService
```

Example 4

Connect to the AD LDS instance on 'server.domain.local:389' with the credentials of the locally logged on user, and create a new AD LDS group in a certain container:

```
C:\PS> new-QADGroup -Service 'server.domain.local:389' -Name 'group1' -ParentContainer
'<DN of container>'
```

Get-QADGroupMember

Retrieve the members of a group in Active Directory. Supported are both Active Directory Domain Services (AD DS) and Active Directory Lightweight Directory Services (AD LDS).

Syntax

```
Get-QADGroupMember [-Identity] <IdentityParameter> [-Indirect] [-Type <String>]
[-PageSize <Int32>] [-SizeLimit <Int32>] [-LdapFilter <String>] [-WildcardMode
<WildcardMode>] [-ObjectAttributes <Object>] [-Description <String>] [-DisplayName
<String>] [-Name <String>] [-Anr <String>] [-CreatedOn <DateTime>] [-CreatedAfter
<DateTime>] [-CreatedBefore <DateTime>] [-LastChangedOn <DateTime>] [-LastChangedAfter
<DateTime>] [-LastChangedBefore <DateTime>] [-IncludeAllProperties]
[-DontConvertValuesToFriendlyRepresentation] [-SerializeValues]
[-ReturnPropertyNamesOnly] [-DontUseDefaultIncludedProperties]
[-UseDefaultExcludedProperties] [-ExcludedProperties <String[]>] [-IncludedProperties
<String[]>] [-UseDefaultExcludedPropertiesExcept <String[]>] [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Anr

Specify a value to be resolved using ambiguous name resolution (ANR). Which attributes are included in an ANR search depends upon the Active Directory schema. Thus, in Windows Server 2003 based Active Directory, the following attributes are set for ANR by default:

- Display-Name (displayName)
- Given-Name (givenName)
- Legacy-Exchange-DN (legacyExchangeDN)
- ms-DS-Additional-Sam-Account-Name (msDS-AdditionalSamAccountName)
- Physical-Delivery-Office-Name (physicalDeliveryOfficeName)
- Proxy-Addresses (proxyAddresses)
- RDN (name)
- SAM-Account-Name (sAMAccountName)
- Surname (sn)

For instance, when you supply 'ann*' as the value of this parameter, the cmdlet searches for objects that have ann at the beginning of the value of at least one of the attributes listed above.

CreatedAfter

Specify the lower boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created after the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedBefore

Specify the upper boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created before the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedOn

Specify the object creation date by which to filter objects found, searching for objects created within the date specified. This parameter is mutually exclusive with the CreatedAfter and CreatedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

Description

Search by the 'description' attribute.

DisplayName

Search by the 'displayName' attribute.

DontConvertValuesToFriendlyRepresentation

This parameter causes the cmdlet to represent the Integer8 and OctetString attribute values “as is,” without converting them to a user-friendly, human-readable form. If this parameter is omitted, the cmdlet performs the following data conversions:

- The values of the Integer8 attributes listed in the Integer8AttributesThatContainDateTimes array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to DateTime.
- The values of the Integer8 attributes listed in the Integer8AttributesThatContainNegativeTimeSpans array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to TimeSpan.
- The values of the other Integer8 attributes are converted from IADsLargeInteger to Int64.
- The values of the OctetString attributes are converted from byte[] to BinHex strings.

Note: This parameter has an effect only on the properties of the output object that have the member type of NoteProperty. Such properties are normally added to the output object in order to provide access to the attribute values of the respective directory object that are loaded to the local memory cache but cannot be accessed by using properties of the base object (the object for which the output object serves as a wrapper).

DontUseDefaultIncludedProperties

This parameter causes the cmdlet to load only a small set of attributes from the directory to the local memory cache (normally, this set is limited to objectClass and ADsPath). Other attributes are retrieved from the directory as needed when you use the cmdlet output objects to read attribute values. Thus, if you want only to count the objects that meet certain conditions (rather than examine values of particular attributes), then you can use this parameter to increase performance of your search. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

ExcludedProperties

Use this parameter to specify the attributes that you do not want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the ExcludedProperties parameter you can change this default behavior on an ad-hoc basis, in order to prevent certain attributes from being loaded. Another scenario involves the use of this parameter in conjunction with IncludeAllProperties in order to restrict the set of the cached attributes.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

IncludeAllProperties

With this parameter, the cmdlet retrieves all attributes of the respective directory object (such as a User object), and stores the attribute values in the memory cache on the local computer. Attribute values can be read from the memory cache by using properties of the object returned by the cmdlet. Thus, when used in conjunction with the SerializeValues parameter, it allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

IncludedProperties

Use this parameter to specify the attributes that you want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the IncludedProperty parameter you can direct the cmdlet to cache some attributes in addition to the default set.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet.

Indirect

Supply this parameter for the cmdlet to retrieve objects that belong to the group because of group nesting, in addition to objects that are direct members of the group.

If this parameter is omitted, the cmdlet retrieves only those objects that are direct members of the group. If this parameter is supplied, the cmdlet takes the immediate list of members of the group and then also recursively expands each group in this list to determine its group memberships to arrive at a complete closed set of the members.

LastChangedAfter

Specify the lower boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed after the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedBefore

Specify the upper boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed before the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedOn

Specify the object modification date by which to filter objects found, searching for objects that have last changed within the date specified. This parameter is mutually exclusive with the LastChangedAfter and LastChangedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

LdapFilter

Specify the LDAP search filter that defines your search criteria on the group members. Note that the search filter string is case-sensitive.

Name

Search by the 'name' attribute.

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet searches for objects that have the specified attributes set to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to search. A value may include an asterisk character - a wildcard representing any group of characters. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

PageSize

Set the maximum number of items in each page of the search results that will be returned by the cmdlet. After the directory server has found the number of objects that are specified by this parameter, it will stop searching and return the results to the cmdlet. When the cmdlet requests more data, the server will restart the search where it left off. You can use this setting to adjust the number of requests (network calls) to the directory server issued by the cmdlet during a search.

Normally, the default page size is 50. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

ReturnPropertyNamesOnly

This parameter causes the cmdlet to list the names of the object attributes whose values the cmdlet retrieves from the directory and stores in the memory cache on the local computer. Thus, when used in conjunction with the IncludeAllProperties parameter, it lists the names of all attributes of the respective directory object (such as a User object). For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

SerializeValues

This parameter causes the cmdlet to output an object whose properties store the attribute values of the respective directory object that are loaded to the local memory cache. The value returned by each property of the output object is represented as a string (serialized) so as to facilitate the export of the attribute values to a text file. Thus, when used in conjunction with the IncludeAllProperties parameter, this parameter allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

SizeLimit

Set the maximum number of items to be returned by the cmdlet. Normally, the default size limit is 1000. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

Type

Specify the type of directory objects to find. The cmdlet searches for objects that have one of the 'objectClass' attribute values set to the Type parameter value.

UseDefaultExcludedProperties

When set to 'true', this parameter causes the cmdlet not to load a certain pre-defined set of attributes from the directory to the local memory cache. This pre-defined set of attributes (referred to as "default excluded properties") can be viewed or modified by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Normally, this parameter is used in conjunction with IncudeAllProperties to avoid retrieval of unnecessary data from the directory server, thereby increasing performance of the search operation performed by the cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

UseDefaultExcludedPropertiesExcept

This parameter is deprecated, and has no effect.

WildcardMode

Specify either 'PowerShell' or 'LDAP' as the parameter value. Normally, if this parameter is not supplied, the cmdlet assumes that WildcardMode is set to 'LDAP'. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. The 'PowerShell' value causes the cmdlet to use PowerShell wildcards and quoting rules. Wildcards are processed on the client side, which may result in slow search performance.

For information about PowerShell wildcards and quoting rules, type the following commands at the PowerShell command-prompt:

```
help about_wildcard
help about_quoting_rule
```

The 'LDAP' value causes the cmdlet to use LDAP wildcards (asterisks only) and LDAP quoting rules (backslash as the escape character). Wildcards are processed on the server side, which enables faster search results.

Detailed Description

Use this cmdlet to retrieve the directory objects that are members of a certain group in Active Directory.

The output of the cmdlet is a collection of objects, with each object representing one of the directory objects found by the cmdlet. You can pipe the output into another cmdlet, such as **Set-QADObject**, to make changes to the directory objects returned by this cmdlet.

Examples

Example 1

Connect to any available domain controller with the credentials of the locally logged on user, bind to a specific group by Domain\Name, and display a list of members of the group:

```
C:\PS> get-QADGroupMember 'MyDomain\Administrators'
```

Example 2

Connect to the AD LDS instance on 'server.domain.local:389' with the credentials of the locally logged on user, bind to a specific AD LDS group by DN, and display a list of members of the group:

```
C:\PS> get-QADGroupMember '<DN of group>' -Service 'server.domain.local:389'
```

Example 3

Retrieve users that belong to a particular group (directly or because of group nesting):

```
C:\PS> Get-QADGroupMember 'domainName\groupName' -Type 'user' -Indirect
```

Add-QADGroupMember

Add one or more objects to a group in Active Directory. Supported are both Active Directory Domain Services (AD DS) and Active Directory Lightweight Directory Services (AD LDS).

Syntax

```
Add-QADGroupMember [-Identity] <IdentityParameter> [-Member] <IdentityParameter[]>
[-Proxy] [-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Confirm

Prompts you for confirmation before executing the command.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

Member

Specify a list of objects to be added to the group. Each list entry is the DN, SID, GUID, UPN or Domain\Name of a directory object. Separate the list entries by commas.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to add objects to a group in Active Directory. You can specify a list of objects to add, separating the list entries by commas.

Examples

Example 1

Connect to the local Administration Service with the credentials of a specific user, add two objects (the first one specified by Domain\Name, the second one specified by SID) to the group, and then disconnect:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-QADService -service 'localhost' -proxy -ConnectionAccount
'company\administrator' -ConnectionPassword $pw
C:\PS> add-QADGroupMember -identity 'CN=group1,OU=companyOU,DC=company,DC=com' -member
'company\jsmith','S-1-5-21-1279736177-1630491018-182859109-1215'
C:\PS> disconnect-QADService
```

Example 2

Connect to the AD LDS instance on 'server.domain.local:389' with the credentials of the locally logged on user, bind to a specific AD LDS group by DN, and add the object with a certain DN to the group:

```
C:\PS> add-QADGroupMember '<DN of group>' -Service 'server.domain.local:389' -Member
'<DN of object>'
```

Remove-QADGroupMember

Remove one or more members from a group in Active Directory. Supported are both Active Directory Domain Services (AD DS) and Active Directory Lightweight Directory Services (AD LDS).

Syntax

```
Remove-QADGroupMember [-Identity] <IdentityParameter> [-Member] <IdentityParameter[]>
[-Proxy] [-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Confirm

Prompts you for confirmation before executing the command.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

Member

Specify a list of objects to be removed from the group. Each list entry is the DN, SID, GUID, UPN or Domain\Name of a directory object. Separate the list entries by commas.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to remove members from a group in Active Directory. You can specify a list of objects to remove, separating the list items by commas.

Examples

Example 1

Connect to the local Administration Service with the credentials of a specific user, remove two members (the first one specified by Domain\AccounrName, the second one specified by SID) from the group, and then disconnect:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-QADService -service 'localhost' -proxy -ConnectionAccount
'company\administrator' -ConnectionPassword $pw
C:\PS> remove-QADGroupMember 'CN=group1,OU=companyOU,DC=company,DC=com' -member
'company\jsmith','S-1-5-21-1279736177-1630491018-182859109-1215'
C:\PS> disconnect-QADService
```

Example 2

Connect to the AD LDS instance on 'server.domain.local:389' with the credentials of the locally logged on user, bind to a specific AD LDS group by DN, and remove the object with a certain DN from the group:

```
C:\PS> remove-QADGroupMember '<DN of group>' -Service 'server.domain.local:389' -Member
'<DN of object>'
```

Get-QADMemberOf

Retrieve group memberships of a particular object in Active Directory. Supported are both Active Directory Domain Services (AD DS) and Active Directory Lightweight Directory Services (AD LDS).

Syntax

```
Get-QADMemberOf [-Identity] <IdentityParameter> [-Indirect] [-PageSize <Int32>]
[-SizeLimit <Int32>] [-LdapFilter <String>] [-WildcardMode <WildcardMode>]
[-ObjectAttributes <Object>] [-Description <String>] [-DisplayName <String>] [-Name
<String>] [-Anr <String>] [-CreatedOn <DateTime>] [-CreatedAfter <DateTime>]
[-CreatedBefore <DateTime>] [-LastChangedOn <DateTime>] [-LastChangedAfter <DateTime>]
[-LastChangedBefore <DateTime>] [-IncludeAllProperties]
[-DontConvertValuesToFriendlyRepresentation] [-SerializeValues]
[-ReturnPropertyNamesOnly] [-DontUseDefaultIncludedProperties]
[-UseDefaultExcludedProperties] [-ExcludedProperties <String[]>] [-IncludedProperties
<String[]>] [-UseDefaultExcludedPropertiesExcept <String[]>] [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Anr

Specify a value to be resolved using ambiguous name resolution (ANR). Which attributes are included in an ANR search depends upon the Active Directory schema. Thus, in Windows Server 2003 based Active Directory, the following attributes are set for ANR by default:

- Display-Name (displayName)
- Given-Name (givenName)
- Legacy-Exchange-DN (legacyExchangeDN)
- ms-DS-Additional-Sam-Account-Name (msDS-AdditionalSamAccountName)
- Physical-Delivery-Office-Name (physicalDeliveryOfficeName)
- Proxy-Addresses (proxyAddresses)
- RDN (name)
- SAM-Account-Name (sAMAccountName)
- Surname (sn)

For instance, when you supply 'ann*' as the value of this parameter, the cmdlet searches for objects that have ann at the beginning of the value of at least one of the attributes listed above.

CreatedAfter

Specify the lower boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created after the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedBefore

Specify the upper boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created before the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedOn

Specify the object creation date by which to filter objects found, searching for objects created within the date specified. This parameter is mutually exclusive with the CreatedAfter and CreatedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

Description

Search by the 'description' attribute.

DisplayName

Search by the 'displayName' attribute.

DontConvertValuesToFriendlyRepresentation

This parameter causes the cmdlet to represent the Integer8 and OctetString attribute values “as is,” without converting them to a user-friendly, human-readable form. If this parameter is omitted, the cmdlet performs the following data conversions:

- The values of the Integer8 attributes listed in the Integer8AttributesThatContainDateTimes array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to DateTime.
- The values of the Integer8 attributes listed in the Integer8AttributesThatContainNegativeTimeSpans array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to TimeSpan.
- The values of the other Integer8 attributes are converted from IADsLargeInteger to Int64.
- The values of the OctetString attributes are converted from byte[] to BinHex strings.

Note: This parameter has an effect only on the properties of the output object that have the member type of NoteProperty. Such properties are normally added to the output object in order to provide access to the attribute values of the respective directory object that are loaded to the local memory cache but cannot be accessed by using properties of the base object (the object for which the output object serves as a wrapper).

DontUseDefaultIncludedProperties

This parameter causes the cmdlet to load only a small set of attributes from the directory to the local memory cache (normally, this set is limited to objectClass and ADsPath). Other attributes are retrieved from the directory as needed when you use the cmdlet output objects to read attribute values. Thus, if you want only to count the objects that meet certain conditions (rather than examine values of particular attributes), then you can use this parameter to increase performance of your search. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

ExcludedProperties

Use this parameter to specify the attributes that you do not want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the ExcludedProperties parameter you can change this default behavior on an ad-hoc basis, in order to prevent certain attributes from being loaded. Another scenario involves the use of this parameter in conjunction with IncludeAllProperties in order to restrict the set of the cached attributes.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

Identity

Specify the DN, SID, GUID, or Domain\Name of the object whose group memberships you want the cmdlet to retrieve. When you pipe an object into the cmdlet, this parameter is used to receive the object.

IncludeAllProperties

With this parameter, the cmdlet retrieves all attributes of the respective directory object (such as a User object), and stores the attribute values in the memory cache on the local computer. Attribute values can be read from the memory cache by using properties of the object returned by the cmdlet. Thus, when used in conjunction with the SerializeValues parameter, it allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

IncludedProperties

Use this parameter to specify the attributes that you want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the IncludedProperty parameter you can direct the cmdlet to cache some attributes in addition to the default set.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet.

Indirect

Supply this parameter on the command line if you want the cmdlet to retrieve groups to which the object specified by the Identity parameter belongs because of group nesting, in addition to groups of which the object is a direct member.

If this parameter is omitted, the cmdlet retrieves only those groups of which the object is a direct member. If this parameter is supplied, the cmdlet takes the immediate group membership list of the object and then also recursively expands each group in this list to determine its group memberships to arrive at a complete set of the groups.

LastChangedAfter

Specify the lower boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed after the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedBefore

Specify the upper boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed before the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedOn

Specify the object modification date by which to filter objects found, searching for objects that have last changed within the date specified. This parameter is mutually exclusive with the LastChangedAfter and LastChangedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

LdapFilter

Specify the LDAP search filter that defines your search criteria. Note that the search filter string is case-sensitive.

The cmdlet disregards this parameter if an Identity value is supplied. If you want this parameter to have effect, do not supply an Identity parameter value.

If you supply the LdapFilter parameter along with attribute-specific parameters, then your search returns objects that meet the conditions defined by the LDAP filter and have the specified attributes set to the specified values.

Name

Search by the 'name' attribute.

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet searches for objects that have the specified attributes set to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to search. A value may include an asterisk character - a wildcard representing any group of characters. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

PageSize

Set the maximum number of items in each page of the search results that will be returned by the cmdlet. After the directory server has found the number of objects that are specified by this parameter, it will stop searching and return the results to the cmdlet. When the cmdlet requests more data, the server will restart the search where it left off. You can use this setting to adjust the number of requests (network calls) to the directory server issued by the cmdlet during a search.

Normally, the default page size is 50. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

ReturnPropertyNamesOnly

This parameter causes the cmdlet to list the names of the object attributes whose values the cmdlet retrieves from the directory and stores in the memory cache on the local computer. Thus, when used in conjunction with the IncludeAllProperties parameter, it lists the names of all attributes of the respective directory object (such as a User object). For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

SerializeValues

This parameter causes the cmdlet to output an object whose properties store the attribute values of the respective directory object that are loaded to the local memory cache. The value returned by each property of the output object is represented as a string (serialized) so as to facilitate the export of the attribute values to a text file. Thus, when used in conjunction with the IncludeAllProperties parameter, this parameter allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

SizeLimit

Set the maximum number of items to be returned by the cmdlet. Normally, the default size limit is 1000. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

UseDefaultExcludedProperties

When set to 'true', this parameter causes the cmdlet not to load a certain pre-defined set of attributes from the directory to the local memory cache. This pre-defined set of attributes (referred to as "default excluded properties") can be viewed or modified by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Normally, this parameter is used in conjunction with IncudeAllProperties to avoid retrieval of unnecessary data from the directory server, thereby increasing performance of the search operation performed by the cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

UseDefaultExcludedPropertiesExcept

This parameter is deprecated, and has no effect.

WildcardMode

Specify either 'PowerShell' or 'LDAP' as the parameter value. Normally, if this parameter is not supplied, the cmdlet assumes that WildcardMode is set to 'LDAP'. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. The 'PowerShell' value causes the cmdlet to use PowerShell wildcards and quoting rules. Wildcards are processed on the client side, which may result in slow search performance.

For information about PowerShell wildcards and quoting rules, type the following commands at the PowerShell command-prompt:

```
help about_wildcard
help about_quoting_rule
```

The 'LDAP' value causes the cmdlet to use LDAP wildcards (asterisks only) and LDAP quoting rules (backslash as the escape character). Wildcards are processed on the server side, which enables faster search results.

Detailed Description

Use this cmdlet to retrieve the groups to which a particular object belongs in Active Directory. You can pass an object or one of its identity-related attributes to the Identity parameter and have the cmdlet return the object's groups that match the search conditions specified.

The output of the cmdlet is a collection of objects, with each object representing one of the groups found by the cmdlet. You can pipe the output into another cmdlet, such as Set-QADGroup, to make changes to the group objects returned by this cmdlet.

Examples

Example 1

Retrieve groups of which a particular user is a direct member:

```
C:\PS> Get-QADMemberOf 'domainName\userName'
```

Example 2

Retrieve groups whose names begin with DL.Administration and where a particular user has membership whether directly or because of group nesting:

```
C:\PS> Get-QADMemberOf 'domainName\userName' -Indirect -Name 'DL.Administration*'
```

Get-QADComputer

Retrieve all computer objects in a domain or container that match the specified conditions.

Syntax

```
Get-QADComputer [[-Identity] <IdentityParameter>] [-ComputerRole <ComputerRole>]
[-ManagedBy <IdentityParameter>] [-SamAccountName <String>] [-DnsName <String>]
[-Location <String>] [-OSName <String>] [-OSVersion <String>] [-OSServicePack <String>]
[-MemberOf <IdentityParameter[]>] [-IndirectMemberOf <IdentityParameter[]>]
[-NotMemberOf <IdentityParameter[]>] [-NotIndirectMemberOf <IdentityParameter[]>]
[-Tombstone] [-LastKnownParent <IdentityParameter>] [-SecurityMask <SecurityMasks>]
[-SearchRoot <IdentityParameter>] [-SearchScope <SearchScope>] [-AttributeScopeQuery
<String>] [-PageSize <Int32>] [-SizeLimit <Int32>] [-LdapFilter <String>] [-WildcardMode
<WildcardMode>] [-ObjectAttributes <Object>] [-Description <String>] [-DisplayName
<String>] [-Name <String>] [-Anr <String>] [-CreatedOn <DateTime>] [-CreatedAfter
<DateTime>] [-CreatedBefore <DateTime>] [-LastChangedOn <DateTime>] [-LastChangedAfter
<DateTime>] [-LastChangedBefore <DateTime>] [-IncludeAllProperties]
[-DontConvertValuesToFriendlyRepresentation] [-SerializeValues]
[-ReturnPropertyNamesOnly] [-DontUseDefaultIncludedProperties]
[-UseDefaultExcludedProperties] [-ExcludedProperties <String[]>] [-IncludedProperties
<String[]>] [-UseDefaultExcludedPropertiesExcept <String[]>] [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>]
```

The cmdlet has a number of optional parameters for searching by individual attributes in the directory, with each parameter name identifying a certain attribute that you can search for attribute values specified by using the respective parameter (see the list of parameters for this cmdlet).

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Anr

Specify a value to be resolved using ambiguous name resolution (ANR). Which attributes are included in an ANR search depends upon the Active Directory schema. Thus, in Windows Server 2003 based Active Directory, the following attributes are set for ANR by default:

- Display-Name (displayName)
- Given-Name (givenName)
- Legacy-Exchange-DN (legacyExchangeDN)
- ms-DS-Additional-Sam-Account-Name (msDS-AdditionalSamAccountName)
- Physical-Delivery-Office-Name (physicalDeliveryOfficeName)

- Proxy-Addresses (proxyAddresses)
- RDN (name)
- SAM-Account-Name (sAMAccountName)
- Surname (sn)

For instance, when you supply 'ann*' as the value of this parameter, the cmdlet searches for objects that have ann at the beginning of the value of at least one of the attributes listed above.

AttributeScopeQuery

Specify the LDAP display name of an attribute that has DN syntax (for example, "member" or "memberOf"). The cmdlet enumerates the Distinguished Name values of the attribute on the object specified by the SearchRoot parameter, and performs the search on the objects represented by the Distinguished Names. The SearchScope parameter has no effect in this case. The object to search must be specified by using the SearchRoot parameter rather than the Identity parameter.

For instance, with the value of this parameter set to "memberOf", the cmdlet searches the collection of the groups to which the SearchRoot object belongs.

ComputerRole

Specify one of these parameter values: 'Member' (to search for computers that are not domain controllers) or 'DomainController' (to search for domain controllers only). If this parameter is omitted, the cmdlet searches for both domain controllers and computers that are not domain controllers.

CreatedAfter

Specify the lower boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created after the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedBefore

Specify the upper boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created before the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedOn

Specify the object creation date by which to filter objects found, searching for objects created within the date specified. This parameter is mutually exclusive with the CreatedAfter and CreatedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

DontConvertValuesToFriendlyRepresentation

This parameter causes the cmdlet to represent the Integer8 and OctetString attribute values "as is," without converting them to a user-friendly, human-readable form. If this parameter is omitted, the cmdlet performs the following data conversions:

- The values of the Integer8 attributes listed in the Integer8AttributesThatContainDateTimes array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to DateTime.
- The values of the Integer8 attributes listed in the Integer8AttributesThatContainNegativeTimeSpans array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to TimeSpan.
- The values of the other Integer8 attributes are converted from IADsLargeInteger to Int64.
- The values of the OctetString attributes are converted from byte[] to BinHex strings.

Note: This parameter has an effect only on the properties of the output object that have the member type of NoteProperty. Such properties are normally added to the output object in order to provide access to the attribute values of the respective directory object that are loaded to the local memory cache but cannot be accessed by using properties of the base object (the object for which the output object serves as a wrapper).

DontUseDefaultIncludedProperties

This parameter causes the cmdlet to load only a small set of attributes from the directory to the local memory cache (normally, this set is limited to objectClass and ADsPath). Other attributes are retrieved from the directory as needed when you use the cmdlet output objects to read attribute values. Thus, if you want only to count the objects that meet certain conditions (rather than examine values of particular attributes), then you can use this parameter to increase performance of your search. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

ExcludedProperties

Use this parameter to specify the attributes that you do not want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the ExcludedProperties parameter you can change this default behavior on an ad-hoc basis, in order to prevent certain attributes from being loaded. Another scenario involves the use of this parameter in conjunction with IncludeAllProperties in order to restrict the set of the cached attributes.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to find.

The cmdlet attempts to retrieve the object identified by this parameter value, disregarding the other search parameters. If you want other search parameters to have effect, do not supply this parameter.

IncludeAllProperties

With this parameter, the cmdlet retrieves all attributes of the respective directory object (such as a User object), and stores the attribute values in the memory cache on the local computer. Attribute values can be read from the memory cache by using properties of the object returned by the cmdlet. Thus, when used in conjunction with the SerializeValues parameter, it allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

IncludedProperties

Use this parameter to specify the attributes that you want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the IncludedProperty parameter you can direct the cmdlet to cache some attributes in addition to the default set.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet.

IndirectMemberOf

Retrieve objects that belong to the group or groups specified by this parameter, whether directly or because of group nesting. The cmdlet returns an object if the object has direct or indirect membership in the group specified by this parameter value.

LastChangedAfter

Specify the lower boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed after the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedBefore

Specify the upper boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed before the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedOn

Specify the object modification date by which to filter objects found, searching for objects that have last changed within the date specified. This parameter is mutually exclusive with the LastChangedAfter and LastChangedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

LastKnownParent

When searching for a deleted object by using the Tombstone parameter, specify the DN of the container the object was in before it became a tombstone. This allows you to find objects that were deleted from a particular container.

Note that the lastKnownParent attribute is only set if the object was deleted on a domain controller running Windows Server 2003 or later version of Microsoft Windows Server. Therefore, it is possible that the lastKnownParent attribute value is inaccurate.

LdapFilter

Specify the LDAP search filter that defines your search criteria. Note that the search filter string is case-sensitive.

The cmdlet disregards this parameter if an Identity value is supplied. If you want this parameter to have effect, do not supply an Identity parameter value.

If you supply the LdapFilter parameter along with attribute-specific parameters, then your search returns objects that meet the conditions defined by the LDAP filter and have the specified attributes set to the specified values.

MemberOf

Retrieve objects that are direct members of the group or groups specified by this parameter. The cmdlet returns an object if the object has direct membership in the group specified by this parameter value.

NotIndirectMemberOf

Retrieve objects that do not belong to the group or groups specified by this parameter, whether directly or because of group nesting. The cmdlet returns an object if the object has neither direct nor indirect membership in the group specified by this parameter value.

NotMemberOf

Retrieve objects that are not direct members of the group or groups specified by this parameter. The cmdlet returns an object if the object does not have direct membership in the group specified by this parameter value.

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet searches for objects that have the specified attributes set to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to search. A value may include an asterisk character - a wildcard representing any group of characters. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

PageSize

Set the maximum number of items in each page of the search results that will be returned by the cmdlet. After the directory server has found the number of objects that are specified by this parameter, it will stop searching and return the results to the cmdlet. When the cmdlet requests more data, the server will restart the search where it left off. You can use this setting to adjust the number of requests (network calls) to the directory server issued by the cmdlet during a search.

Normally, the default page size is 50. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

ReturnPropertyNamesOnly

This parameter causes the cmdlet to list the names of the object attributes whose values the cmdlet retrieves from the directory and stores in the memory cache on the local computer. Thus, when used in conjunction with the IncludeAllProperties parameter, it lists the names of all attributes of the respective directory object (such as a User object). For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

SearchRoot

Specify the Distinguished Name, Canonical Name, or GUID of the container to search. By default, the cmdlet searches the entire sub-tree of which SearchRoot is the topmost object (sub-tree search). This default behavior can be altered by using the SearchScope parameter.

The search criteria are defined by the LdapFilter parameter value and the values of attribute-specific parameters.

The cmdlet disregards this parameter if an Identity value is supplied. If you want this parameter to have effect, do not supply any Identity parameter value.

SearchScope

Specify one of these parameter values:

- 'Base' - Limits the search to the base (SearchRoot) object. The result contains a maximum of one object.
- 'OneLevel' - Searches the immediate descendant (child) objects of the base (SearchRoot) object, excluding the base object.
- 'Subtree' - Searches the whole sub-tree, including the base (SearchRoot) object and all its descendant objects.

Normally, if this parameter is not supplied, the cmdlet performs a Subtree search. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

SecurityMask

Specify which elements of the object’s security descriptor to retrieve. Valid parameter values are:

- 'None' - do not retrieve any security data
- 'Owner' - retrieve the owner data
- 'Group' - retrieve the primary group data
- 'Dacl' - retrieve the discretionary access-control list data
- 'Sacl' - retrieve the system access-control list data

You can supply a combination of these values, separating them by commas. For example, you can supply the parameter value of 'Dacl,Sacl' in order to retrieve both the discretionary and system access-control list data.

SerializeValues

This parameter causes the cmdlet to output an object whose properties store the attribute values of the respective directory object that are loaded to the local memory cache. The value returned by each property of the output object is represented as a string (serialized) so as to facilitate the export of the attribute values to a text file. Thus, when used in conjunction with the IncludeAllProperties parameter, this parameter allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

SizeLimit

Set the maximum number of items to be returned by the cmdlet. Normally, the default size limit is 1000. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

Tombstone

Search for deleted objects of the respective object class. The search output is normally intended to be passed (piped in) to the Restore-QADDeletedObject cmdlet for restoring deleted objects.

When Active Directory deletes an object from the directory, the object becomes a tombstone. A search using this parameter returns tombstone objects that meet the filtering criteria supplied. Note that upon deletion of an object only a small number of the object's attributes are saved in the tombstone, with most of the attributes being lost. To search for deleted objects, your search filter should be based on the attributes that are preserved in tombstones. Which attributes are preserved is determined by Active Directory. Thus, the attributes that have the 0x8 bit set in the searchFlags attribute of the attributeSchema definition are preserved.

When the Tombstone parameter is supplied, the search results include deleted objects (tombstones) that match the specified search filter. However, a search filter that matches a live object may not work as expected after the object is deleted. This is because not all attributes are retained in the tombstone. For example, a filter such as (&(objectClass=user)(objectCategory=person)) would not match any tombstone objects since the objectCategory attribute is removed upon object deletion. Conversely, the objectClass attribute is retained on tombstone objects, so a filter of (objectClass=user) would match deleted user objects.

The name (RDN) of a tombstone object begins with the name (RDN) of the deleted object, so a search using the Tombstone parameter can be refined by adding a filter based on object name. For example, to search for deleted objects with a name that begins with "John", you can use a filter such as (cn=John*).

It is also possible to find a specific deleted object. If you know the name (RDN) of the object and the Distinguished Name (DN) of the container the object was in before it was deleted, then you can pass the container's DN to the LastKnownParent parameter and apply a filter of (cn=<RDN of the object>*) in order to have the cmdlet retrieve that specific object. However, if an object is deleted, a new object with the same DN is created, and then deleted as well, the above search would return more than one tombstone object. The returned tombstone objects are distinguished by the GUIDs of the deleted objects, with the name of each tombstone object ending in the GUID of the respective deleted object.

UseDefaultExcludedProperties

When set to 'true', this parameter causes the cmdlet not to load a certain pre-defined set of attributes from the directory to the local memory cache. This pre-defined set of attributes (referred to as "default excluded properties") can be viewed or modified by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Normally, this parameter is used in conjunction with IncudeAllProperties to avoid retrieval of unnecessary data from the directory server, thereby increasing performance of the search operation performed by the cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

UseDefaultExcludedPropertiesExcept

This parameter is deprecated, and has no effect.

WildcardMode

Specify either 'PowerShell' or 'LDAP' as the parameter value. Normally, if this parameter is not supplied, the cmdlet assumes that WildcardMode is set to 'LDAP'. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. The 'PowerShell' value causes the cmdlet to use PowerShell wildcards and quoting rules. Wildcards are processed on the client side, which may result in slow search performance.

For information about PowerShell wildcards and quoting rules, type the following commands at the PowerShell command-prompt:

```
help about_wildcard
help about_quoting_rule
```

The 'LDAP' value causes the cmdlet to use LDAP wildcards (asterisks only) and LDAP quoting rules (backslash as the escape character). Wildcards are processed on the server side, which enables faster search results.

Attribute-specific Parameters

The cmdlet also takes a series of optional, attribute-specific parameters allowing you to search by computer attributes. The attribute-specific parameters have effect if SearchRoot is specified whereas Identity is not. If you specify SearchRoot only, then the cmdlet returns all computer objects found in the SearchRoot container.

You can use attribute-specific parameters to search for computer objects that have specific values of certain attributes. With more than one attribute-specific parameter supplied, the search conditions are combined by using the AND operator, so as to find the computer objects that meet all the specified conditions.

If a particular attribute is referred to by both the ObjectAttributes array and the attribute-specific parameter, the ObjectAttributes setting has no effect on that attribute. The cmdlet searches for the attribute value specified by the attribute-specific parameter.

Each of the attribute-specific parameters accepts the asterisk (*) wildcard character in the parameter value to match zero or more characters (case-insensitive).

The following table lists the attribute-specific parameters you can use with this cmdlet. Each parameter adds a filter condition based on a certain attribute identified by the LDAP display name in the table.

TO SEARCH BY THIS ATTRIBUTE...	USE THIS SYNTAX
description	-Description <String>
displayName	-DisplayName <String>
samAccountName	-SamAccountName <String>
dNSHostName	-DnsName <String>
location	-Location <String>
managedBy	-ManagedBy <String>
operatingSystem	-OSName <String>
operatingSystemVersion	-OSVersion <String>
operatingSystemServicePack	-OSServicePack <String>

Detailed Description

Use this cmdlet to search an Active Directory domain or container for computer objects that meet certain criteria, or to bind to a certain computer object by DN, SID, GUID, or Domain\Name. You can search by computer attributes or specify your search criteria by using an LDAP search filter.

The output of the cmdlet is a collection of objects, with each object representing one of the computer objects found by the cmdlet. You can pipe the output into another cmdlet, such as **Set-QADObject**, to make changes to the computer objects returned by this cmdlet.

Examples

Example 1

Bind to a particular computer by Domain\Name, and display the computer's name and DN. In this example, the NetBIOS name of the domain is assumed to be "MyDomain" and the pre-Windows 2000 name of the computer is assumed to be "MyServer":

```
C:\PS> get-QADComputer 'MyDomain\MyServer$'
```

Example 2

With a specific OU, find all computers that run a particular version of the operating system, and list the names of the computers found. The OU is identified by its canonical name.

```
C:\PS> get-QADComputer -SearchRoot 'company.com/computersOU' -OSName '*Vista*'
```

Example 3

Find all domain controllers in your domain, and list their names and DNs:

```
C:\PS> get-QADComputer -computerRole 'DomainController'
```

Example 4

Find all computers in your domain; for each computer found, display the pre-Windows 2000 computer name along with the operating system name, version, and service pack:

```
C:\PS> Get-QADComputer | format-table -property computername, osname, osversion,
osservicepack
```

This command displays the computers in a table with four columns: "ComputerName", "OSName", "OSVersion" and "OSServicePack." The command uses the **Get-QADComputer** cmdlet to get all of the computers. The pipeline operator (|) sends the results to the **Format-Table** cmdlet, which formats the output in a table. The *Property* parameter specifies the properties that appear in the table as columns.

Note: ComputerName, OSName, OSVersion and OSServicePack are just four of the properties of an object returned by the **Get-QADComputer** cmdlet. To see all of the properties, type the following command:

```
C:\PS> get-qadcomputer | get-member
```

Example 5

Connect to a specific domain controller with the credentials of a specific user, and bind to a certain computer account by Domain\Name, display the computer name and description, and then disconnect:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-QADService -service 'server.company.com' -ConnectionAccount
'company\administrator' -ConnectionPassword $pw
C:\PS> get-QADComputer 'company\computer$' | ft computername, description
C:\PS> disconnect-QADService
```

Example 6

Connect to any available domain controller with the credentials of the locally logged on user, search for computers in a specific container by using an LDAP search filter, and display the name and DN of each computer found:

```
C:\PS> get-QADComputer -SearchRoot 'company.com/ComputersOU' -LdapFilter
'(description=a*)'
```

Example 7

Connect to any available domain controller with the credentials of a specific user, search a certain container to find all computers with empty description, set a description for each of those computers, and then disconnect:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-QADService -ConnectionAccount 'company\administrator'
-ConnectionPassword $pw
C:\PS> get-QADComputer -SearchRoot 'company.com/ComputersOU' -description '' |
set-QADObject -description 'A description'
C:\PS> disconnect-QADService
```

Get-QADObject

Retrieve all directory objects in a domain or container that match the specified conditions. Supported are both Active Directory Domain Services (AD DS) and Active Directory Lightweight Directory Services (AD LDS).

Syntax

```
Get-QADObject [[-Identity] <IdentityParameter>] [-Type <String>] [-MemberOf
<IdentityParameter[]>] [-IndirectMemberOf <IdentityParameter[]>] [-NotMemberOf
<IdentityParameter[]>] [-NotIndirectMemberOf <IdentityParameter[]>] [-Tombstone]
[-LastKnownParent <IdentityParameter>] [-SecurityMask <SecurityMasks>] [-SearchRoot
<IdentityParameter>] [-SearchScope <SearchScope>] [-AttributeScopeQuery <String>]
[-PageSize <Int32>] [-SizeLimit <Int32>] [-LdapFilter <String>] [-WildcardMode
<WildcardMode>] [-ObjectAttributes <Object>] [-Description <String>] [-DisplayName
<String>] [-Name <String>] [-Anr <String>] [-CreatedOn <DateTime>] [-CreatedAfter
<DateTime>] [-CreatedBefore <DateTime>] [-LastChangedOn <DateTime>] [-LastChangedAfter
<DateTime>] [-LastChangedBefore <DateTime>] [-IncludeAllProperties]
[-DontConvertValuesToFriendlyRepresentation] [-SerializeValues]
[-ReturnPropertyNamesOnly] [-DontUseDefaultIncludedProperties]
[-UseDefaultExcludedProperties] [-ExcludedProperties <String[]>] [-IncludedProperties
<String[]>] [-UseDefaultExcludedPropertiesExcept <String[]>] [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Anr

Specify a value to be resolved using ambiguous name resolution (ANR). Which attributes are included in an ANR search depends upon the Active Directory schema. Thus, in Windows Server 2003 based Active Directory, the following attributes are set for ANR by default:

- Display-Name (displayName)
- Given-Name (givenName)
- Legacy-Exchange-DN (legacyExchangeDN)
- ms-DS-Additional-Sam-Account-Name (msDS-AdditionalSamAccountName)
- Physical-Delivery-Office-Name (physicalDeliveryOfficeName)
- Proxy-Addresses (proxyAddresses)
- RDN (name)

- SAM-Account-Name (sAMAccountName)
- Surname (sn)

For instance, when you supply 'ann*' as the value of this parameter, the cmdlet searches for objects that have ann at the beginning of the value of at least one of the attributes listed above.

AttributeScopeQuery

Specify the LDAP display name of an attribute that has DN syntax (for example, "member" or "memberOf"). The cmdlet enumerates the Distinguished Name values of the attribute on the object specified by the SearchRoot parameter, and performs the search on the objects represented by the Distinguished Names. The SearchScope parameter has no effect in this case. The object to search must be specified by using the SearchRoot parameter rather than the Identity parameter.

For instance, with the value of this parameter set to "memberOf", the cmdlet searches the collection of the groups to which the SearchRoot object belongs.

CreatedAfter

Specify the lower boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created after the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedBefore

Specify the upper boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created before the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedOn

Specify the object creation date by which to filter objects found, searching for objects created within the date specified. This parameter is mutually exclusive with the CreatedAfter and CreatedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

Description

Search by the 'description' attribute.

DisplayName

Search by the 'displayName' attribute.

DontConvertValuesToFriendlyRepresentation

This parameter causes the cmdlet to represent the Integer8 and OctetString attribute values “as is,” without converting them to a user-friendly, human-readable form. If this parameter is omitted, the cmdlet performs the following data conversions:

- The values of the Integer8 attributes listed in the Integer8AttributesThatContainDateTimes array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to DateTime.

- The values of the Integer8 attributes listed in the Integer8AttributesThatContainNegativeTimeSpans array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to TimeSpan.
- The values of the other Integer8 attributes are converted from IADsLargeInteger to Int64.
- The values of the OctetString attributes are converted from byte[] to BinHex strings.

Note: This parameter has an effect only on the properties of the output object that have the member type of NoteProperty. Such properties are normally added to the output object in order to provide access to the attribute values of the respective directory object that are loaded to the local memory cache but cannot be accessed by using properties of the base object (the object for which the output object serves as a wrapper).

DontUseDefaultIncludedProperties

This parameter causes the cmdlet to load only a small set of attributes from the directory to the local memory cache (normally, this set is limited to objectClass and ADsPath). Other attributes are retrieved from the directory as needed when you use the cmdlet output objects to read attribute values. Thus, if you want only to count the objects that meet certain conditions (rather than examine values of particular attributes), then you can use this parameter to increase performance of your search. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

ExcludedProperties

Use this parameter to specify the attributes that you do not want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the ExcludedProperties parameter you can change this default behavior on an ad-hoc basis, in order to prevent certain attributes from being loaded. Another scenario involves the use of this parameter in conjunction with IncludeAllProperties in order to restrict the set of the cached attributes.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to find.

The cmdlet attempts to retrieve the object identified by this parameter value, disregarding the other search parameters. If you want other search parameters to have effect, do not supply this parameter.

IncludeAllProperties

With this parameter, the cmdlet retrieves all attributes of the respective directory object (such as a User object), and stores the attribute values in the memory cache on the local computer. Attribute values can be read from the memory cache by using properties of the object returned by the cmdlet. Thus, when used in conjunction with the SerializeValues parameter, it allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

IncludedProperties

Use this parameter to specify the attributes that you want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the IncludedProperty parameter you can direct the cmdlet to cache some attributes in addition to the default set.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet.

IndirectMemberOf

Retrieve objects that belong to the group or groups specified by this parameter, whether directly or because of group nesting. The cmdlet returns an object if the object has direct or indirect membership in the group specified by this parameter value.

LastChangedAfter

Specify the lower boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed after the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedBefore

Specify the upper boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed before the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedOn

Specify the object modification date by which to filter objects found, searching for objects that have last changed within the date specified. This parameter is mutually exclusive with the LastChangedAfter and LastChangedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

LastKnownParent

When searching for a deleted object by using the Tombstone parameter, specify the DN of the container the object was in before it became a tombstone. This allows you to find objects that were deleted from a particular container.

Note that the lastKnownParent attribute is only set if the object was deleted on a domain controller running Windows Server 2003 or later version of Microsoft Windows Server. Therefore, it is possible that the lastKnownParent attribute value is inaccurate.

LdapFilter

Specify the LDAP search filter that defines your search criteria. Note that the search filter string is case-sensitive.

The cmdlet disregards this parameter if an Identity value is supplied. If you want this parameter to have effect, do not supply an Identity parameter value.

If you supply the LdapFilter parameter along with attribute-specific parameters, then your search returns objects that meet the conditions defined by the LDAP filter and have the specified attributes set to the specified values.

MemberOf

Retrieve objects that are direct members of the group or groups specified by this parameter. The cmdlet returns an object if the object has direct membership in the group specified by this parameter value.

Name

Search by the 'name' attribute.

NotIndirectMemberOf

Retrieve objects that do not belong to the group or groups specified by this parameter, whether directly or because of group nesting. The cmdlet returns an object if the object has neither direct nor indirect membership in the group specified by this parameter value.

NotMemberOf

Retrieve objects that are not direct members of the group or groups specified by this parameter. The cmdlet returns an object if the object does not have direct membership in the group specified by this parameter value.

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet searches for objects that have the specified attributes set to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to search. A value may include an asterisk character - a wildcard representing any group of characters. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

PageSize

Set the maximum number of items in each page of the search results that will be returned by the cmdlet. After the directory server has found the number of objects that are specified by this parameter, it will stop searching and return the results to the cmdlet. When the cmdlet requests more data, the server will restart the search where it left off. You can use this setting to adjust the number of requests (network calls) to the directory server issued by the cmdlet during a search.

Normally, the default page size is 50. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

ReturnPropertyNamesOnly

This parameter causes the cmdlet to list the names of the object attributes whose values the cmdlet retrieves from the directory and stores in the memory cache on the local computer. Thus, when used in conjunction with the IncludeAllProperties parameter, it lists the names of all attributes of the respective directory object (such as a User object). For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

SearchRoot

Specify the Distinguished Name, Canonical Name, or GUID of the container to search. By default, the cmdlet searches the entire sub-tree of which SearchRoot is the topmost object (sub-tree search). This default behavior can be altered by using the SearchScope parameter.

The search criteria are defined by the LdapFilter parameter value and the values of attribute-specific parameters.

The cmdlet disregards this parameter if an Identity value is supplied. If you want this parameter to have effect, do not supply any Identity parameter value.

SearchScope

Specify one of these parameter values:

- 'Base' - Limits the search to the base (SearchRoot) object. The result contains a maximum of one object.
- 'OneLevel' - Searches the immediate descendant (child) objects of the base (SearchRoot) object, excluding the base object.
- 'Subtree' - Searches the whole sub-tree, including the base (SearchRoot) object and all its descendant objects.

Normally, if this parameter is not supplied, the cmdlet performs a Subtree search. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

SecurityMask

Specify which elements of the object’s security descriptor to retrieve. Valid parameter values are:

- 'None' - do not retrieve any security data
- 'Owner' - retrieve the owner data
- 'Group' - retrieve the primary group data
- 'Dacl' - retrieve the discretionary access-control list data
- 'Sacl' - retrieve the system access-control list data

You can supply a combination of these values, separating them by commas. For example, you can supply the parameter value of 'Dacl,Sacl' in order to retrieve both the discretionary and system access-control list data.

SerializeValues

This parameter causes the cmdlet to output an object whose properties store the attribute values of the respective directory object that are loaded to the local memory cache. The value returned by each property of the output object is represented as a string (serialized) so as to facilitate the export of the attribute values to a text file. Thus, when used in conjunction with the IncludeAllProperties parameter, this parameter allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

SizeLimit

Set the maximum number of items to be returned by the cmdlet. Normally, the default size limit is 1000. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

Tombstone

Search for deleted objects of the respective object class. The search output is normally intended to be passed (piped in) to the Restore-QADDeletedObject cmdlet for restoring deleted objects.

When Active Directory deletes an object from the directory, the object becomes a tombstone. A search using this parameter returns tombstone objects that meet the filtering criteria supplied. Note that upon deletion of an object only a small number of the object's attributes are saved in the tombstone, with most of the attributes being lost. To search for deleted objects, your search filter should be based on the attributes that are preserved in tombstones. Which attributes are preserved is determined by Active Directory. Thus, the attributes that have the 0x8 bit set in the searchFlags attribute of the attributeSchema definition are preserved.

When the Tombstone parameter is supplied, the search results include deleted objects (tombstones) that match the specified search filter. However, a search filter that matches a live object may not work as expected after the object is deleted. This is because not all attributes are retained in the tombstone. For example, a filter such as (&(objectClass=user)(objectCategory=person)) would not match any tombstone objects since the objectCategory attribute is removed upon object deletion. Conversely, the objectClass attribute is retained on tombstone objects, so a filter of (objectClass=user) would match deleted user objects.

The name (RDN) of a tombstone object begins with the name (RDN) of the deleted object, so a search using the Tombstone parameter can be refined by adding a filter based on object name. For example, to search for deleted objects with a name that begins with "John", you can use a filter such as (cn=John*).

It is also possible to find a specific deleted object. If you know the name (RDN) of the object and the Distinguished Name (DN) of the container the object was in before it was deleted, then you can pass the container's DN to the LastKnownParent parameter and apply a filter of (cn=<RDN of the object>*) in order to have the cmdlet retrieve that specific object. However, if an object is deleted, a new object with the same DN is created, and then deleted as well, the above search would return more than one tombstone object. The returned tombstone objects are distinguished by the GUIDs of the deleted objects, with the name of each tombstone object ending in the GUID of the respective deleted object.

Type

Specify the type of directory objects to find. The cmdlet searches for objects that have one of the 'objectClass' attribute values set to the Type parameter value.

UseDefaultExcludedProperties

When set to 'true', this parameter causes the cmdlet not to load a certain pre-defined set of attributes from the directory to the local memory cache. This pre-defined set of attributes (referred to as "default excluded properties") can be viewed or modified by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Normally, this parameter is used in conjunction with IncudeAllProperties to avoid retrieval of unnecessary data from the directory server, thereby increasing performance of the search operation performed by the cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

UseDefaultExcludedPropertiesExcept

This parameter is deprecated, and has no effect.

WildcardMode

Specify either 'PowerShell' or 'LDAP' as the parameter value. Normally, if this parameter is not supplied, the cmdlet assumes that WildcardMode is set to 'LDAP'. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. The 'PowerShell' value causes the cmdlet to use PowerShell wildcards and quoting rules. Wildcards are processed on the client side, which may result in slow search performance.

For information about PowerShell wildcards and quoting rules, type the following commands at the PowerShell command-prompt:

```
help about_wildcard
help about_quoting_rule
```

The 'LDAP' value causes the cmdlet to use LDAP wildcards (asterisks only) and LDAP quoting rules (backslash as the escape character). Wildcards are processed on the server side, which enables faster search results.

Detailed Description

Use this cmdlet to search an Active Directory domain or container for directory objects that meet certain criteria, or to bind to a certain object by DN, SID, GUID, UPN, or Domain\Name. You can search by object attributes or specify your search criteria by using an LDAP search filter.

The output of the cmdlet is a collection of objects, with each object representing one of the directory objects found by the cmdlet. You can pipe the output into another cmdlet, such as **Set-QADObject**, to make changes to the directory objects returned by this cmdlet.

Examples

Example 1

Connect to any available domain controller with the credentials of the locally logged on user, bind to a specific object by DN, and display the description of the object:

```
C:\PS> (get-QADObject 'CN=John
Smith,OU=CompanyOU,DC=company,DC=com').DirectoryEntry.description
```

Example 2

Connect to a specific domain controller with the credentials of a specific user, bind to a certain object by SID, display the description of the object, and then disconnect:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-QADService -service 'server.company.com' -ConnectionAccount
'company\administrator' -ConnectionPassword $pw
C:\PS> (get-QADObject -identity
'S-1-5-21-1279736177-1630491018-182859109-1305').DirectoryEntry.description
C:\PS> disconnect-QADService
```

Example 3

Connect to any available domain controller with the credentials of the locally logged on user, search for objects in a specific container by using an LDAP search filter, and display a list of the objects found:

```
C:\PS> get-QADObject -SearchRoot 'company.com/UsersOU' -LdapFilter '(description=a*)'
```

Example 4

Connect to any available domain controller with the credentials of the locally logged on user, find all computer objects in a specific container, and display a list of the objects found:

```
C:\PS> get-QADObject -SearchRoot 'company.com/ComputersOU' -Type Computer
```

Example 5

Connect to any available domain controller with the credentials of a specific user, search a certain container to find all groups with the empty Notes field, set a note for each of those groups, and then disconnect:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-QADService -ConnectionAccount 'company\administrator'
-ConnectionPassword $pw
C:\PS> get-QADObject -SearchRoot 'company.com/GroupsOU' -Type Group –ObjectAttributes
@{info=''} | set-QADObject -ObjectAttributes @{info='A note'}
C:\PS> disconnect-QADService
```

Example 6

List the names of all properties of organizationalUnit objects. Sort the list by property name:

```
C:\PS> get-QADObject -Type 'organizationalUnit' -IncludeAllProperties
-ReturnPropertyNamesOnly | ForEach-Object {$_} | Sort-Object
```

Example 7

Connect to the AD LDS instance on 'server.domain.local:389' with the credentials of the locally logged on user, find all AD LDS objects in a specific container, and display a list of the objects found:

```
C:\PS> get-QADObject -Service 'server.domain.local:389' -SearchRoot '<DN of container>'
```

Example 8

Retrieve all objects that were deleted on the current date:

```
C:\PS> Get-QADObject -Tombstone -LastChangedOn (get-date)
```

Set-QADObject

Modify attributes of an object in Active Directory. Supported are both Active Directory Domain Services (AD DS) and Active Directory Lightweight Directory Services (AD LDS).

Syntax

```
Set-QADObject [[-Identity] <IdentityParameter>] [-ObjectAttributes
<ObjectAttributesParameter>] [-Description <String>] [-DisplayName <String>]
[-ExcludedProperties <String[]>] [-IncludedProperties <String[]>] [-DeserializeValues]
[-UseDefaultExcludedProperties] [-UseDefaultExcludedPropertiesExcept <String[]>]
[-Proxy] [-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Confirm

Prompts you for confirmation before executing the command.

Description

Set the 'description' attribute to this parameter value.

DeserializeValues

Supply this parameter if the input you pass to the cmdlet contains serialized attribute values (for instance, when importing a directory object from a text file that was created using the Serialize parameter). For examples of how to export and import an object, see documentation on the Get-QADUser cmdlet.

DisplayName

Set the 'displayName' attribute to this parameter value.

ExcludedProperties

Use this parameter to specify the attributes that you do not want the cmdlet to set in the directory. Supply a list of the attribute LDAP display names as the parameter value. You could use this parameter when importing attribute values from a text file, in order to prevent some attributes found in the file from being set in the directory.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

IncludedProperties

Use this parameter to specify explicitly the attributes that you want the cmdlet to set in the directory. Supply a list of the attribute LDAP display names as the parameter value. When used together with UseDefaultExcludedProperties, this parameter allows you to have the cmdlet set some attributes that would not be set otherwise.

Note: If a particular attribute is listed in both ExcludedProperties and IncludedProperties, the cmdlet does not set the value of that attribute in the directory.

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet sets the specified attributes to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to set. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

UseDefaultExcludedProperties

When set to 'true', this parameter causes the cmdlet not to make changes to certain attributes in the directory. This pre-defined set of attributes (referred to as "default excluded properties") can be viewed or modified by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

UseDefaultExcludedPropertiesExcept

This parameter is deprecated, and has no effect.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to change or remove values of attributes of an object in Active Directory.

Examples

Example 1

Connect to any available domain controller with the credentials of the locally logged on user, bind to a specific object by DN, and modify the description of the object:

```
C:\PS> set-QADObject 'CN=John Smith,OU=CompanyOU,DC=company,DC=com' -description 'Sales person'
```

Example 2

Connect to a specific domain controller with the credentials of a specific user, bind to a certain object by SID, modify the description of the object, and then disconnect:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-QADService -service 'server.company.com' -ConnectionAccount
'company\administrator' -ConnectionPassword $pw
C:\PS> set-QADObject -identity 'S-1-5-21-1279736177-1630491018-182859109-1305'
-description 'Service account'
C:\PS> disconnect-QADService
```

Example 3

Connect to the local Administration Service with the credentials of a specific user, bind to a certain object by Domain\Name, set or clear certain attributes, and then disconnect:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-QADService -service 'localhost' -proxy -ConnectionAccount
'company\administrator' -ConnectionPassword $pw
C:\PS> set-QADObject -identity 'company\associates' -ObjectAttributes
@{info='';description='All company associates'}
C:\PS> disconnect-QADService
```

Example 4

Connect to the AD LDS instance on 'server.domain.local:389' with the credentials of the locally logged on user, bind to a specific AD LDS object by DN, and modify the description of the object:

```
C:\PS> set-QADObject '<DN of object>' -Service 'server.domain.local:389' -description
'My AD LDS object'
```

New-QADObject

Create a new object of in Active Directory. Supported are both Active Directory Domain Services (AD DS) and Active Directory Lightweight Directory Services (AD LDS).

Syntax

```
New-QADObject [-Name] <String> -ParentContainer <IdentityParameter> -Type <String>
[-NamingProperty <String>] [-ObjectAttributes <Object>] [-Description <String>]
[-DisplayName <String>] [-ExcludedProperties <String[]>] [-IncludedProperties
<String[]>] [-DeserializeValues] [-UseDefaultExcludedProperties]
[-UseDefaultExcludedPropertiesExcept <String[]>] [-Proxy] [-UseGlobalCatalog] [-Service
<String>] [-ConnectionAccount <String>] [-ConnectionPassword <SecureString>]
[-Credential <PSCredential>] [-Connection <ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Confirm

Prompts you for confirmation before executing the command.

Description

Set the 'description' attribute to this parameter value.

DeserializeValues

Supply this parameter if the input you pass to the cmdlet contains serialized attribute values (for instance, when importing a directory object from a text file that was created using the Serialize parameter). For examples of how to export and import an object, see documentation on the Get-QADUser cmdlet.

DisplayName

Set the 'displayName' attribute to this parameter value.

ExcludedProperties

Use this parameter to specify the attributes that you do not want the cmdlet to set in the directory. Supply a list of the attribute LDAP display names as the parameter value. You could use this parameter when importing attribute values from a text file, in order to prevent some attributes found in the file from being set in the directory.

IncludedProperties

Use this parameter to specify explicitly the attributes that you want the cmdlet to set in the directory. Supply a list of the attribute LDAP display names as the parameter value. When used together with UseDefaultExcludedProperties, this parameter allows you to have the cmdlet set some attributes that would not be set otherwise.

Note: If a particular attribute is listed in both ExcludedProperties and IncludedProperties, the cmdlet does not set the value of that attribute in the directory.

Name

Set the 'name' attribute to this parameter value on the new object created by this cmdlet in the directory.

NamingProperty

Supply the LDAP name of the naming attribute specific to the object class of the directory object you want to create. The naming attribute qualifies the object's relative Distinguished Name. If this parameter is omitted, the naming attribute is assumed to be 'cn', which is suitable for most object classes. Other possible values are 'ou' (naming attribute of the organizationalUnit object class) and 'dc' (naming attribute of the domain object class).

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet sets the specified attributes to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to set. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

ParentContainer

Specify the Distinguished Name of the container in which you want the new directory object to be created by this cmdlet.

Type

Specify the object class of the directory object to be created. This is the name of a schema class object, such as User or Group. The cmdlet creates a directory object of the object class specified by the value of this parameter.

UseDefaultExcludedProperties

When set to 'true', this parameter causes the cmdlet not to make changes to certain attributes in the directory. This pre-defined set of attributes (referred to as "default excluded properties") can be viewed or modified by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

UseDefaultExcludedPropertiesExcept

This parameter is deprecated, and has no effect.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to create a directory object of the specified schema class and a particular name in the container. The cmdlet also allows for setting other properties (for example, the mandatory properties) on the new object.

Examples

Example 1

Connect to any available domain controller with the credentials of the locally logged on user, and create a new computer object:

```
C:\PS> new-qadObject -ParentContainer 'OU=ComputersOU,DC=company,DC=com' -type
'computer' -name 'comp1' -ObjectAttributes @{sAMAccountName='comp1'}
```

Example 2

Connect to the local Administration Service with the credentials of a specific user, create a new organizational unit, and then disconnect:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-qadService -service 'localhost' -proxy             -ConnectionAccount
'company\administrator' -ConnectionPassword $pw
C:\PS> new-qadObject -ParentContainer 'OU=companyOU,DC=company,DC=com' -type
'organizationalUnit'       -NamingProperty 'ou' -name 'Child OU'
C:\PS> disconnect-qadService
```

Example 3

Connect to the AD LDS instance on 'server.domain.local:389' with the credentials of the locally logged on user, and create a new AD LDS user object in a certain container:

```
C:\PS> new-QADObject -Service 'server.domain.local:389' -ParentContainer '<DN of
container>' -Type 'user' -Name 'John Smith'
```

Move-QADObject

Move the specified object to a different location (container) in Active Directory.

Syntax

```
Move-QADObject [-Identity] <IdentityParameter> -NewParentContainer <IdentityParameter>
[-Proxy] [-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Confirm

Prompts you for confirmation before executing the command.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

NewParentContainer

Specify the Distinguished Name of the destination container (the container to which you want to move the object).

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to move an object between containers within an Active Directory domain (the cmdlet cannot move an object to a different domain). An object to move can be specified by DN, SID, GUID, UPN or Domain\Name, or it can be located by using a **Get-** cmdlet and then piped into the **Move-** cmdlet. The destination container can be specified by DN or GUID.

Examples

Example 1

Connect to any available domain controller with the credentials of the locally logged on user, bind to a particular user object by Domain\Name, and move the object to the specified container:

```
C:\PS> move-QADObject 'MyDomain\JSmith' -NewParentContainer
'MyDomain.company.com/NewYork/Users'
```

Rename-QADObject

Change the name of the specified object in Active Directory.

Syntax

```
Rename-QADObject [-Identity] <IdentityParameter> -NewName <String> [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Confirm

Prompts you for confirmation before executing the command.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

NewName

Specify the new name to assign to the directory object (set the 'name' attribute to this parameter value).

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to rename an object in Active Directory. The cmdlet sets the *name* attribute of the object to the value specified, thus causing the corresponding change to the distinguished name of the object. An object to rename can be specified by DN, SID, GUID, UPN or Domain\Name, or it can be located by using a **Get-** cmdlet and then piped into the **Rename-** cmdlet.

Examples

Example 1

Connect to any available domain controller with the credentials of the locally logged on user, bind to a specific user object by Domain\Name, and assign the new name to the object:

```
C:\PS> rename-QADObject 'MyDomain\JSmith' -NewName 'Jane Smith'
```

Remove-QADObject

Delete the specified objects in Active Directory. Supported are both Active Directory Domain Services (AD DS) and Active Directory Lightweight Directory Services (AD LDS).

Syntax

```
Remove-QADObject [-Identity] <IdentityParameter> [-DeleteTree] [-Force] [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Confirm

Prompts you for confirmation before executing the command.

DeleteTree

Deletes the specified object along with all of its descendant objects (the entire sub-tree). If you omit this parameter, the cmdlet fails to delete container objects that hold any child objects.

Force

Overrides restrictions that prevent the command from succeeding, just so the changes do not compromise security. Suppresses the warning or confirmation messages that could appear during changes caused by execution of the command. This parameter is useful when you run the command programmatically, so interactive input is inappropriate.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to delete objects in Active Directory. An object to delete can be specified by DN, SID, GUID, UPN or Domain\Name, or it can be located by using a **Get-** cmdlet and then piped into the **Remove-** cmdlet (see examples).

Examples

Example 1

Connect to any available domain controller with the credentials of the locally logged on user, bind to a specific user object by Domain\Name, and delete the object:

```
C:\PS> remove-QADObject 'MyDomain\JSmith'
```

Example 2

Connect to any available domain controller with the credentials of the locally logged on user, and delete all user objects that are located in a specific container:

```
C:\PS> get-QADUser -searchRoot 'mydomain.company.com/usersOU' | remove-QADObject
-confirm
```

Example 3

Connect to the local Administration Service with the credentials of a specific user, and delete a certain container along with all objects that are located in that container:

```
C:\PS> $pw = read-host "Enter password" -AsSecureString
C:\PS> connect-QADService -service 'localhost' -proxy -ConnectionAccount
'company\administrator' -ConnectionPassword $pw
C:\PS> remove-QADObject 'mydomain.company.com/usersOU' -deleteTree -force
C:\PS> disconnect-QADService
```

Example 4

Connect to the AD LDS instance on 'server.domain.local:389' with the credentials of the locally logged on user, bind to a specific AD LDS object by DN, and delete the object:

```
C:\PS> remove-QADObject '<DN of object>' -service 'server.domain.local:389' -confirm
```

Restore-QADDeletedObject

Undelete objects in Active Directory by restoring tombstones back into normal objects. This cmdlet requires an Active Directory domain controller running Windows Server 2003 or later.

Syntax

```
Restore-QADDeletedObject -TargetObject <ArsDirectoryObject> [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Confirm

Prompts you for confirmation before executing the command.

TargetObject

Supply tombstone objects to restore. This parameter is normally used to receive the output of a Get-QAD cmdlet searching for deleted objects (see Tombstone parameter on Get-QAD* cmdlets).

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to recover deleted objects in Active Directory. Pipe a tombstone object into this cmdlet in order to undelete the object represented by the tombstone. Tombstone objects can be retrieved by using an appropriate Get-QAD* cmdlet with the Tombstone parameter.

When an object is deleted, it is not physically removed from the Active Directory database. Instead, Active Directory marks the object as deleted, clears most of the object's attributes, renames the object and moves it to a special container. The object becomes a tombstone. This cmdlet causes Active Directory to perform the necessary operations on the tombstone to reanimate the object, which effectively results in the object being undeleted. When undeleted, the object appears in the LostAndFound container of the respective domain.

The object's attributes that were cleared upon object deletion are not restored. However, certain attributes, the most important of which are identity-related attributes such as objectGUID and objectSid, are retained in the tombstone, and restored when the object is undeleted. Which attributes are retained in tombstones is determined by Active Directory. Thus, the attributes that have the 0x8 bit set in the searchFlags attribute of the attributeSchema definition are retained.

The cmdlet relies on the "restore deleted objects" feature of Active Directory. To enable this feature, at least one domain controller in the domain must be running on Windows Server 2003 or a later version of Microsoft Windows Server. Ensure that the cmdlet is connected to such a domain controller. Normally, only domain administrators are allowed to restore deleted objects. For information about access rights required to restore deleted objects and limitations that apply to restoring deleted objects, see topic "Restoring Deleted Objects" in MSDN Library at http://msdn.microsoft.com.

Examples

Example 1

Restore a user account that was deleted from a particular container and had the name (RDN) of John Smith:

```
C:\PS> Get-QADUser -Tombstone -LastKnownParent '<DN of container>' –Name 'John Smith*' |
Restore-QADDeletedObject
```

Example 2

Restore all user accounts that were deleted from a particular container on the current date:

```
C:\PS> Get-QADUser –Tombstone –LastKnownParent '<DN of container>' -LastChangedOn
(get-date) | Restore-QADDeletedObject
```

Example 3

Restore all user accounts that were deleted on September 1, 2008:

```
C:\PS> Get-QADUser –Tombstone –LastChangedOn (get-date -year 2008 -month 9 -day 1) |
Restore-QADDeletedObject
```

New-QADPasswordSettingsObject

Create a new Password Settings object (PSO). Active Directory version of Windows Server 2008 or later is required.

Syntax

```
New-QADPasswordSettingsObject [-Name] <String> [-ParentContainer <IdentityParameter>]
[-AppliesTo <IdentityParameter[]>] [-Precedence <Int32>] [-ReversibleEncryptionEnabled]
[-PasswordHistoryLength <Int32>] [-PasswordComplexityEnabled] [-MinimumPasswordLength
<Int32>] [-MinimumPasswordAge <Object>] [-MaximumPasswordAge <Object>]
[-LockoutThreshold <Int32>] [-ResetLockoutCounterAfter <Object>] [-LockoutDuration
<Object>] [-ObjectAttributes <ObjectAttributesParameter>] [-Description <String>]
[-DisplayName <String>] [-ExcludedProperties <String[]>] [-IncludedProperties
<String[]>] [-DeserializeValues] [-UseDefaultExcludedProperties]
[-UseDefaultExcludedPropertiesExcept <String[]>] [-Proxy] [-UseGlobalCatalog] [-Service
<String>] [-ConnectionAccount <String>] [-ConnectionPassword <SecureString>]
[-Credential <PSCredential>] [-Connection <ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

AppliesTo

Specify a list of users or groups to which you want the Password Settings object to apply. Each list entry is the DN, SID, GUID, UPN or Domain\Name of a user or group. Separate the list entries by commas.

Confirm

Prompts you for confirmation before executing the command.

Description

Set the 'description' attribute to this parameter value.

DeserializeValues

Supply this parameter if the input you pass to the cmdlet contains serialized attribute values (for instance, when importing a directory object from a text file that was created using the Serialize parameter). For examples of how to export and import an object, see documentation on the Get-QADUser cmdlet.

DisplayName

Set the 'displayName' attribute to this parameter value.

ExcludedProperties

Use this parameter to specify the attributes that you do not want the cmdlet to set in the directory. Supply a list of the attribute LDAP display names as the parameter value. You could use this parameter when importing attribute values from a text file, in order to prevent some attributes found in the file from being set in the directory.

IncludedProperties

Use this parameter to specify explicitly the attributes that you want the cmdlet to set in the directory. Supply a list of the attribute LDAP display names as the parameter value. When used together with UseDefaultExcludedProperties, this parameter allows you to have the cmdlet set some attributes that would not be set otherwise.

Note: If a particular attribute is listed in both ExcludedProperties and IncludedProperties, the cmdlet does not set the value of that attribute in the directory.

LockoutDuration

Specify the Lockout Duration setting for locked out user accounts (set the 'msDS-LockoutDuration' attribute to this parameter value).

Parameter value can be represented as any of the following: Int64, IADsLargeInteger, DateTime, TimeSpan, string (a string representation of Int64, DateTime or TimeSpan), or Int (a number of minutes). This must be a negative value (see examples).

LockoutThreshold

Specify the Lockout Threshold setting for lockout of user accounts (set the 'msDS-LockoutThreshold' attribute to this parameter value).

MaximumPasswordAge

Specify the Maximum Password Age setting for user accounts (set the 'msDS-MaximumPasswordAge' attribute to this parameter value).

Parameter value can be represented as any of the following: Int64, IADsLargeInteger, DateTime, TimeSpan, string (a string representation of Int64, DateTime or TimeSpan), or Int (a number of days). This must be a negative value (see examples).

MinimumPasswordAge

Specify the Minimum Password Age setting for user accounts (set the 'msDS-MinimumPasswordAge' attribute to this parameter value).

Parameter value can be represented as any of the following: Int64, IADsLargeInteger, DateTime, TimeSpan, string (a string representation of Int64, DateTime or TimeSpan), or Int (a number of days). This must be a negative value (see examples).

MinimumPasswordLength

Specify the Minimum Password Length setting for user accounts (set the 'msDS-MinimumPasswordLength' attribute to this parameter value).

Name

Set the 'name' attribute to this parameter value on the new object created by this cmdlet in the directory.

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet sets the specified attributes to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to set. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

ParentContainer

Specify the Distinguished Name of the container in which you want the new directory object to be created by this cmdlet.

PasswordComplexityEnabled

Specify either 'true' or 'false' to determine the password complexity status for user accounts (set the 'msDS-PasswordComplexityEnabled' attribute to this parameter value).

PasswordHistoryLength

Specify the Password History Length setting for user accounts (set the 'msDS-PasswordHistoryLength' attribute to this parameter value).

Precedence

Specify the password settings precedence for the Password Settings object (set the 'msDS-PasswordSettingsPrecedence' attribute to this parameter value).

ResetLockoutCounterAfter

Specify the Observation Window setting for lockout of user accounts (set the 'msDS-LockoutObservationWindow' attribute to this parameter value).

Parameter value can be represented as any of the following: Int64, IADsLargeInteger, DateTime, TimeSpan, string (a string representation of Int64, DateTime or TimeSpan), or Int (a number of minutes). This must be a negative value (see examples).

ReversibleEncryptionEnabled

Specify either 'true' or 'false' to determine the password reversible encryption status for user accounts (set the 'msDS-PasswordReversibleEncryptionEnabled' attribute to this parameter value).

UseDefaultExcludedProperties

When set to 'true', this parameter causes the cmdlet not to make changes to certain attributes in the directory. This pre-defined set of attributes (referred to as "default excluded properties") can be viewed or modified by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

UseDefaultExcludedPropertiesExcept

This parameter is deprecated, and has no effect.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to create a Password Settings object (PSO) and set attribute values in the newly created object.

This cmdlet takes a series of attribute-specific parameters allowing you to set attributes in the newly created Password Settings object. If a particular attribute is referred to by both the ObjectAttributes array and an attribute-specific parameter, the ObjectAttributes setting has no effect on that attribute. The cmdlet sets the attribute to the value specified by the attribute-specific parameter.

Examples

Example 1

Create a new PSO named myPso1 with LockoutDuration of 40 min, Precedence of 10 and MaximumPasswordAge of 45 days, 3 hours and 23 minutes and default values for the other parameters, and apply it to two groups, and display operation results:

```
C:\PS> New-QADPasswordSettingsObject -Name 'myPso1' -LockoutDuration 40 -Precedence 10
-MaximumPasswordAge (new-timespan -days -45 -hour -3 -minute -23) -AppliesTo
'myDomain\Account Operators','myDomain\Event Log Readers' | Format-List
```

Get-QADPasswordSettingsObject

Retrieve Password Settings objects that match the specified conditions. Active Directory version of Windows Server 2008 or later is required.

Syntax

```
Get-QADPasswordSettingsObject [[-Identity] <IdentityParameter>] [-Precedence <Int32>]
[-ReversibleEncryptionEnabled] [-PasswordHistoryLength <Int32>]
[-PasswordComplexityEnabled] [-MinimumPasswordLength <Int32>] [-MinimumPasswordAge
<TimeSpanAndDaysParameter>] [-MaximumPasswordAge <TimeSpanAndDaysParameter>]
[-LockoutThreshold <Int32>] [-ResetLockoutCounterAfter <TimeSpanAndMinutesParameter>]
[-LockoutDuration <TimeSpanAndMinutesParameter>] [-MemberOf <IdentityParameter[]>]
[-IndirectMemberOf <IdentityParameter[]>] [-NotMemberOf <IdentityParameter[]>]
[-NotIndirectMemberOf <IdentityParameter[]>] [-Tombstone] [-LastKnownParent
<IdentityParameter>] [-SecurityMask <SecurityMasks>] [-SearchRoot <IdentityParameter>]
[-SearchScope <SearchScope>] [-AttributeScopeQuery <String>] [-PageSize <Int32>]
[-SizeLimit <Int32>] [-LdapFilter <String>] [-WildcardMode <WildcardMode>]
[-ObjectAttributes <Object>] [-Description <String>] [-DisplayName <String>] [-Name
<String>] [-Anr <String>] [-CreatedOn <DateTime>] [-CreatedAfter <DateTime>]
[-CreatedBefore <DateTime>] [-LastChangedOn <DateTime>] [-LastChangedAfter <DateTime>]
[-LastChangedBefore <DateTime>] [-IncludeAllProperties]
[-DontConvertValuesToFriendlyRepresentation] [-SerializeValues]
[-ReturnPropertyNamesOnly] [-DontUseDefaultIncludedProperties]
[-UseDefaultExcludedProperties] [-ExcludedProperties <String[]>] [-IncludedProperties
<String[]>] [-UseDefaultExcludedPropertiesExcept <String[]>] [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Anr

Specify a value to be resolved using ambiguous name resolution (ANR). Which attributes are included in an ANR search depends upon the Active Directory schema. Thus, in Windows Server 2003 based Active Directory, the following attributes are set for ANR by default:

- Display-Name (displayName)
- Given-Name (givenName)
- Legacy-Exchange-DN (legacyExchangeDN)
- ms-DS-Additional-Sam-Account-Name (msDS-AdditionalSamAccountName)
- Physical-Delivery-Office-Name (physicalDeliveryOfficeName)

- Proxy-Addresses (proxyAddresses)
- RDN (name)
- SAM-Account-Name (sAMAccountName)
- Surname (sn)

For instance, when you supply 'ann*' as the value of this parameter, the cmdlet searches for objects that have ann at the beginning of the value of at least one of the attributes listed above.

AttributeScopeQuery

Specify the LDAP display name of an attribute that has DN syntax (for example, 'msDS-PSOAppliesTo'). The cmdlet enumerates the Distinguished Name values of the attribute on the object specified by the SearchRoot parameter, and performs the search on the objects represented by the Distinguished Names. The SearchScope parameter has no effect in this case. The object to search must be specified by using the SearchRoot parameter rather than the Identity parameter.

For instance, with the value of this parameter set to 'msDS-PSOAppliesTo', the cmdlet searches the collection of the users and security groups to which the SearchRoot object is applied.

CreatedAfter

Specify the lower boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created after the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedBefore

Specify the upper boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created before the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedOn

Specify the object creation date by which to filter objects found, searching for objects created within the date specified. This parameter is mutually exclusive with the CreatedAfter and CreatedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

Description

Search by the 'description' attribute.

DisplayName

Search by the 'displayName' attribute.

DontConvertValuesToFriendlyRepresentation

This parameter causes the cmdlet to represent the Integer8 and OctetString attribute values "as is," without converting them to a user-friendly, human-readable form. If this parameter is omitted, the cmdlet performs the following data conversions:

- The values of the Integer8 attributes listed in the Integer8AttributesThatContainDateTimes array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to DateTime.
- The values of the Integer8 attributes listed in the Integer8AttributesThatContainNegativeTimeSpans array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to TimeSpan.
- The values of the other Integer8 attributes are converted from IADsLargeInteger to Int64.
- The values of the OctetString attributes are converted from byte[] to BinHex strings.

Note: This parameter has an effect only on the properties of the output object that have the member type of NoteProperty. Such properties are normally added to the output object in order to provide access to the attribute values of the respective directory object that are loaded to the local memory cache but cannot be accessed by using properties of the base object (the object for which the output object serves as a wrapper).

DontUseDefaultIncludedProperties

This parameter causes the cmdlet to load only a small set of attributes from the directory to the local memory cache (normally, this set is limited to objectClass and ADsPath). Other attributes are retrieved from the directory as needed when you use the cmdlet output objects to read attribute values. Thus, if you want only to count the objects that meet certain conditions (rather than examine values of particular attributes), then you can use this parameter to increase performance of your search. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

ExcludedProperties

Use this parameter to specify the attributes that you do not want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the ExcludedProperties parameter you can change this default behavior on an ad-hoc basis, in order to prevent certain attributes from being loaded. Another scenario involves the use of this parameter in conjunction with IncludeAllProperties in order to restrict the set of the cached attributes.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to find.

The cmdlet attempts to retrieve the object identified by this parameter value, disregarding the other search parameters. If you want other search parameters to have effect, do not supply this parameter.

IncludeAllProperties

With this parameter, the cmdlet retrieves all attributes of the respective directory object (such as a User object), and stores the attribute values in the memory cache on the local computer. Attribute values can be read from the memory cache by using properties of the object returned by the cmdlet. Thus, when used in conjunction with the SerializeValues parameter, it allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

IncludedProperties

Use this parameter to specify the attributes that you want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the IncludedProperty parameter you can direct the cmdlet to cache some attributes in addition to the default set.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet.

IndirectMemberOf

Retrieve objects that belong to the group or groups specified by this parameter, whether directly or because of group nesting. The cmdlet returns an object if the object has direct or indirect membership in the group specified by this parameter value.

LastChangedAfter

Specify the lower boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed after the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedBefore

Specify the upper boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed before the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedOn

Specify the object modification date by which to filter objects found, searching for objects that have last changed within the date specified. This parameter is mutually exclusive with the LastChangedAfter and LastChangedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

LastKnownParent

When searching for a deleted object by using the Tombstone parameter, specify the DN of the container the object was in before it became a tombstone. This allows you to find objects that were deleted from a particular container.

Note that the lastKnownParent attribute is only set if the object was deleted on a domain controller running Windows Server 2003 or later version of Microsoft Windows Server. Therefore, it is possible that the lastKnownParent attribute value is inaccurate.

LdapFilter

Specify the LDAP search filter that defines your search criteria. Note that the search filter string is case-sensitive.

The cmdlet disregards this parameter if an Identity value is supplied. If you want this parameter to have effect, do not supply an Identity parameter value.

If you supply the LdapFilter parameter along with attribute-specific parameters, then your search returns objects that meet the conditions defined by the LDAP filter and have the specified attributes set to the specified values.

LockoutDuration

Specify the Lockout Duration setting of Password Settings objects to find (search by the 'msDS-LockoutDuration' attribute).

Parameter value can be represented as any of the following: Int64, IADsLargeInteger, DateTime, TimeSpan, string (a string representation of Int64, DateTime or TimeSpan), or Int (a number of minutes). This must be a negative value.

LockoutThreshold

Specify the Lockout Threshold setting of Password Settings objects to find (search by the 'msDS-LockoutThreshold' attribute).

MaximumPasswordAge

Specify the Maximum Password Age setting of the objects to find (search by the 'msDS-MaximumPasswordAge' attribute).

Parameter value can be represented as any of the following: Int64, IADsLargeInteger, DateTime, TimeSpan, string (a string representation of Int64, DateTime or TimeSpan), or Int (a number of days). This must be a negative value.

MemberOf

Retrieve objects that are direct members of the group or groups specified by this parameter. The cmdlet returns an object if the object has direct membership in the group specified by this parameter value.

MinimumPasswordAge

Specify the Minimum Password Age setting of Password Settings objects to find (search by the 'msDS-MinimumPasswordAge' attribute).

Parameter value can be represented as any of the following: Int64, IADsLargeInteger, DateTime, TimeSpan, string (a string representation of Int64, DateTime or TimeSpan), or Int (a number of days). This must be a negative value.

MinimumPasswordLength

Specify the Minimum Password Length setting of Password Settings objects to find (search by the 'msDS-MinimumPasswordLength' attribute).

Name

Search by the 'name' attribute.

NotIndirectMemberOf

Retrieve objects that do not belong to the group or groups specified by this parameter, whether directly or because of group nesting. The cmdlet returns an object if the object has neither direct nor indirect membership in the group specified by this parameter value.

NotMemberOf

Retrieve objects that are not direct members of the group or groups specified by this parameter. The cmdlet returns an object if the object does not have direct membership in the group specified by this parameter value.

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet searches for objects that have the specified attributes set to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to search. A value may include an asterisk character - a wildcard representing any group of characters. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

PageSize

Set the maximum number of items in each page of the search results that will be returned by the cmdlet. After the directory server has found the number of objects that are specified by this parameter, it will stop searching and return the results to the cmdlet. When the cmdlet requests more data, the server will restart the search where it left off. You can use this setting to adjust the number of requests (network calls) to the directory server issued by the cmdlet during a search.

Normally, the default page size is 50. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

PasswordComplexityEnabled

Specify either 'true' or 'false' to find Password Settings objects that enable or disable the password complexity requirements for user accounts (search by the 'msDS-PasswordComplexityEnabled' attribute).

PasswordHistoryLength

Specify the Password History Length setting of Password Settings objects to find (search by the 'msDS-PasswordHistoryLength' attribute).

Precedence

Specify the password settings precedence of Password Settings objects to find (search by the 'msDS-PasswordSettingsPrecedence' attribute).

ResetLockoutCounterAfter

Specify the Observation Window setting of Password Settings objects to find (search by the 'msDS-LockoutObservationWindow' attribute).

Parameter value can be represented as any of the following: Int64, IADsLargeInteger, DateTime, TimeSpan, string (a string representation of Int64, DateTime or TimeSpan), or Int (a number of minutes). This must be a negative value.

ReturnPropertyNamesOnly

This parameter causes the cmdlet to list the names of the object attributes whose values the cmdlet retrieves from the directory and stores in the memory cache on the local computer. Thus, when used in conjunction with the IncludeAllProperties parameter, it lists the names of all attributes of the respective directory object (such as a User object). For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

ReversibleEncryptionEnabled

Specify either 'true' or 'false' to find Password Settings objects that enable or disable password reversible encryption for user accounts (search by the 'msDS-PasswordReversibleEncryptionEnabled' attribute).

SearchRoot

Specify the Distinguished Name, Canonical Name, or GUID of the container to search. By default, the cmdlet searches the entire sub-tree of which SearchRoot is the topmost object (sub-tree search). This default behavior can be altered by using the SearchScope parameter.

The search criteria are defined by the LdapFilter parameter value and the values of attribute-specific parameters.

The cmdlet disregards this parameter if an Identity value is supplied. If you want this parameter to have effect, do not supply any Identity parameter value.

SearchScope

Specify one of these parameter values:

- 'Base' - Limits the search to the base (SearchRoot) object. The result contains a maximum of one object.
- 'OneLevel' - Searches the immediate descendant (child) objects of the base (SearchRoot) object, excluding the base object.
- 'Subtree' - Searches the whole sub-tree, including the base (SearchRoot) object and all its descendant objects.

Normally, if this parameter is not supplied, the cmdlet performs a Subtree search. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

SecurityMask

Specify which elements of the object's security descriptor to retrieve. Valid parameter values are:

- 'None' - do not retrieve any security data
- 'Owner' - retrieve the owner data
- 'Group' - retrieve the primary group data
- 'Dacl' - retrieve the discretionary access-control list data
- 'Sacl' - retrieve the system access-control list data

You can supply a combination of these values, separating them by commas. For example, you can supply the parameter value of 'Dacl,Sacl' in order to retrieve both the discretionary and system access-control list data.

SerializeValues

This parameter causes the cmdlet to output an object whose properties store the attribute values of the respective directory object that are loaded to the local memory cache. The value returned by each property of the output object is represented as a string (serialized) so as to facilitate the export of the attribute values to a text file. Thus, when used in conjunction with the IncludeAllProperties parameter, this parameter allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

SizeLimit

Set the maximum number of items to be returned by the cmdlet. Normally, the default size limit is 1000. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

Tombstone

Search for deleted objects of the respective object class. The search output is normally intended to be passed (piped in) to the Restore-QADDeletedObject cmdlet for restoring deleted objects.

When Active Directory deletes an object from the directory, the object becomes a tombstone. A search using this parameter returns tombstone objects that meet the filtering criteria supplied. Note that upon deletion of an object only a small number of the object's attributes are saved in the tombstone, with most of the attributes being lost. To search for deleted objects, your search filter should be based on the attributes that are preserved in tombstones. Which attributes are preserved is determined by Active Directory. Thus, the attributes that have the 0x8 bit set in the searchFlags attribute of the attributeSchema definition are preserved.

When the Tombstone parameter is supplied, the search results include deleted objects (tombstones) that match the specified search filter. However, a search filter that matches a live object may not work as expected after the object is deleted. This is because not all attributes are retained in the tombstone. For example, a filter such as (&(objectClass=user)(objectCategory=person)) would not match any tombstone objects since the objectCategory attribute is removed upon object deletion. Conversely, the objectClass attribute is retained on tombstone objects, so a filter of (objectClass=user) would match deleted user objects.

The name (RDN) of a tombstone object begins with the name (RDN) of the deleted object, so a search using the Tombstone parameter can be refined by adding a filter based on object name. For example, to search for deleted objects with a name that begins with "John", you can use a filter such as (cn=John*).

It is also possible to find a specific deleted object. If you know the name (RDN) of the object and the Distinguished Name (DN) of the container the object was in before it was deleted, then you can pass the container's DN to the LastKnownParent parameter and apply a filter of (cn=<RDN of the object>*) in order to have the cmdlet retrieve that specific object. However, if an object is deleted, a new object with the same DN is created, and then deleted as well, the above search would return more than one tombstone object. The returned tombstone objects are distinguished by the GUIDs of the deleted objects, with the name of each tombstone object ending in the GUID of the respective deleted object.

UseDefaultExcludedProperties

When set to 'true', this parameter causes the cmdlet not to load a certain pre-defined set of attributes from the directory to the local memory cache. This pre-defined set of attributes (referred to as "default excluded properties") can be viewed or modified by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Normally, this parameter is used in conjunction with IncudeAllProperties to avoid retrieval of unnecessary data from the directory server, thereby increasing performance of the search operation performed by the cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

UseDefaultExcludedPropertiesExcept

This parameter is deprecated, and has no effect.

WildcardMode

Specify either 'PowerShell' or 'LDAP' as the parameter value. Normally, if this parameter is not supplied, the cmdlet assumes that WildcardMode is set to 'LDAP'. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. The 'PowerShell' value causes the cmdlet to use PowerShell wildcards and quoting rules. Wildcards are processed on the client side, which may result in slow search performance.

For information about PowerShell wildcards and quoting rules, type the following commands at the PowerShell command-prompt:

```
help about_wildcard
help about_quoting_rule
```

The 'LDAP' value causes the cmdlet to use LDAP wildcards (asterisks only) and LDAP quoting rules (backslash as the escape character). Wildcards are processed on the server side, which enables faster search results.

Detailed Description

Use this cmdlet to search an Active Directory domain or container for Password Settings objects that meet certain search criteria, or to bind to a certain Password Settings object by DN or GUID. You can search by object attributes or specify your search criteria by using an LDAP search filter.

The output of the cmdlet is a collection of objects, with each object representing one of the objects found by the cmdlet. You can pipe the output into another cmdlet, such as **Set-QADObject**, to make changes to the Password Settings objects returned by this cmdlet.

The cmdlet takes a series of attribute-specific parameters allowing you to search by object attributes. The attribute-specific parameters have effect if SearchRoot is specified whereas Identity is not. If you specify SearchRoot only, then the cmdlet returns all Password Settings objects found in the SearchRoot container.

You can use attribute-specific parameters to search for objects that have specific values of certain attributes. Thus, to find all Password Settings objects that have the password settings precedence set to 1, you may add the following on the command line: "-Precedence 1". To search for Password Settings objects that have a certain attribute not set specify '' (empty string) as the parameter value.

If a particular attribute is referred to by both the ObjectAttributes array and an attribute-specific parameter, the ObjectAttributes setting has no effect on that attribute. The cmdlet searches for the attribute value specified by the attribute-specific parameter.

With more than one attribute-specific parameter supplied, the search conditions are combined by using the AND operator, so as to find the objects that meet all the specified conditions.

Examples

Example 1

Find a Password Settings object by name, and display properties of the object found:

```
C:\PS> Get-QADPasswordSettingsObject -Name 'myPso1' | Format-List
```

Example 2

Find all Password Settings objects in your domain, and, for each object found, list the users and groups that the object is applied to:

```
C:\PS> Get-QADPasswordSettingsObject -IncludedProperties 'msDS-PSOAppliesTo' |
Format-Table Name,'msDS-PSOAppliesTo'
```

Example 3

Query on the 'msDS-PSOApplied' attribute to retrieve and display the distinguished names of the Password Settings objects that are (explicitly) applied to the user object specified:

```
C:\PS> Get-QADUser 'john smith' -DontUseDefaultIncludedProperties -IncludedProperties
'msDS-PSOApplied' | Format-Table 'msDS-PSOApplied'
```

Example 4

Query on the 'msDS-PSOApplied' attribute to retrieve the distinguished names of the Password Settings objects that are (explicitly) applied to the user object specified, and store the names in a variable named $psos:

```
C:\PS> $psos = (Get-QADUser 'john smith'
-DontUseDefaultIncludedProperties -IncludedProperties
'msDS-PSOApplied').'msDS-PSOApplied'
```

Example 5

Query on the 'msDS-ResultantPso' attribute to retrieve the distinguished name of the PSO that ultimately applies to the user specified (based on the RSoP calculation rules). If there is no PSO that applies to the user, either directly or by virtue of group membership, the query returns NULL:

```
C:\PS> Get-QADUser 'john smith' -DontUseDefaultIncludedProperties -IncludedProperties
'msDS-ResultantPso' | Format-Table 'msDS-ResultantPso'
```

Get-QADPasswordSettingsObjectAppliesTo

Retrieve objects to which a particular Password Settings object is applied. Active Directory version of Windows Server 2008 or later is required.

Syntax

```
Get-QADPasswordSettingsObjectAppliesTo [-Identity] <IdentityParameter> [-Type <String>]
[-PageSize <Int32>] [-SizeLimit <Int32>] [-LdapFilter <String>] [-WildcardMode
<WildcardMode>] [-ObjectAttributes <Object>] [-Description <String>] [-DisplayName
<String>] [-Name <String>] [-Anr <String>] [-CreatedOn <DateTime>] [-CreatedAfter
<DateTime>] [-CreatedBefore <DateTime>] [-LastChangedOn <DateTime>] [-LastChangedAfter
<DateTime>] [-LastChangedBefore <DateTime>] [-IncludeAllProperties]
[-DontConvertValuesToFriendlyRepresentation] [-SerializeValues]
[-ReturnPropertyNamesOnly] [-DontUseDefaultIncludedProperties]
[-UseDefaultExcludedProperties] [-ExcludedProperties <String[]>] [-IncludedProperties
<String[]>] [-UseDefaultExcludedPropertiesExcept <String[]>] [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Anr

Specify a value to be resolved using ambiguous name resolution (ANR). Which attributes are included in an ANR search depends upon the Active Directory schema. Thus, in Windows Server 2003 based Active Directory, the following attributes are set for ANR by default:

- Display-Name (displayName)
- Given-Name (givenName)
- Legacy-Exchange-DN (legacyExchangeDN)
- ms-DS-Additional-Sam-Account-Name (msDS-AdditionalSamAccountName)
- Physical-Delivery-Office-Name (physicalDeliveryOfficeName)
- Proxy-Addresses (proxyAddresses)
- RDN (name)
- SAM-Account-Name (sAMAccountName)
- Surname (sn)

For instance, when you supply 'ann*' as the value of this parameter, the cmdlet searches for objects that have ann at the beginning of the value of at least one of the attributes listed above.

CreatedAfter

Specify the lower boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created after the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedBefore

Specify the upper boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created before the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedOn

Specify the object creation date by which to filter objects found, searching for objects created within the date specified. This parameter is mutually exclusive with the CreatedAfter and CreatedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

Description

Search by the 'description' attribute.

DisplayName

Search by the 'displayName' attribute.

DontConvertValuesToFriendlyRepresentation

This parameter causes the cmdlet to represent the Integer8 and OctetString attribute values “as is,” without converting them to a user-friendly, human-readable form. If this parameter is omitted, the cmdlet performs the following data conversions:

- The values of the Integer8 attributes listed in the Integer8AttributesThatContainDateTimes array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to DateTime.
- The values of the Integer8 attributes listed in the Integer8AttributesThatContainNegativeTimeSpans array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to TimeSpan.
- The values of the other Integer8 attributes are converted from IADsLargeInteger to Int64.
- The values of the OctetString attributes are converted from byte[] to BinHex strings.

Note: This parameter has an effect only on the properties of the output object that have the member type of NoteProperty. Such properties are normally added to the output object in order to provide access to the attribute values of the respective directory object that are loaded to the local memory cache but cannot be accessed by using properties of the base object (the object for which the output object serves as a wrapper).

DontUseDefaultIncludedProperties

This parameter causes the cmdlet to load only a small set of attributes from the directory to the local memory cache (normally, this set is limited to objectClass and ADsPath). Other attributes are retrieved from the directory as needed when you use the cmdlet output objects to read attribute values. Thus, if you want only to count the objects that meet certain conditions (rather than examine values of particular attributes), then you can use this parameter to increase performance of your search. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

ExcludedProperties

Use this parameter to specify the attributes that you do not want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the ExcludedProperties parameter you can change this default behavior on an ad-hoc basis, in order to prevent certain attributes from being loaded. Another scenario involves the use of this parameter in conjunction with IncludeAllProperties in order to restrict the set of the cached attributes.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

IncludeAllProperties

With this parameter, the cmdlet retrieves all attributes of the respective directory object (such as a User object), and stores the attribute values in the memory cache on the local computer. Attribute values can be read from the memory cache by using properties of the object returned by the cmdlet. Thus, when used in conjunction with the SerializeValues parameter, it allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

IncludedProperties

Use this parameter to specify the attributes that you want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the IncludedProperty parameter you can direct the cmdlet to cache some attributes in addition to the default set.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet.

LastChangedAfter

Specify the lower boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed after the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedBefore

Specify the upper boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed before the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedOn

Specify the object modification date by which to filter objects found, searching for objects that have last changed within the date specified. This parameter is mutually exclusive with the LastChangedAfter and LastChangedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

LdapFilter

Specify the LDAP search filter that defines your search criteria. Note that the search filter string is case-sensitive.

The cmdlet disregards this parameter if an Identity value is supplied. If you want this parameter to have effect, do not supply an Identity parameter value.

If you supply the LdapFilter parameter along with attribute-specific parameters, then your search returns objects that meet the conditions defined by the LDAP filter and have the specified attributes set to the specified values.

Name

Search by the 'name' attribute.

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet searches for objects that have the specified attributes set to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to search. A value may include an asterisk character - a wildcard representing any group of characters. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

PageSize

Set the maximum number of items in each page of the search results that will be returned by the cmdlet. After the directory server has found the number of objects that are specified by this parameter, it will stop searching and return the results to the cmdlet. When the cmdlet requests more data, the server will restart the search where it left off. You can use this setting to adjust the number of requests (network calls) to the directory server issued by the cmdlet during a search.

Normally, the default page size is 50. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

ReturnPropertyNamesOnly

This parameter causes the cmdlet to list the names of the object attributes whose values the cmdlet retrieves from the directory and stores in the memory cache on the local computer. Thus, when used in conjunction with the IncludeAllProperties parameter, it lists the names of all attributes of the respective directory object (such as a User object). For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

SerializeValues

This parameter causes the cmdlet to output an object whose properties store the attribute values of the respective directory object that are loaded to the local memory cache. The value returned by each property of the output object is represented as a string (serialized) so as to facilitate the export of the attribute values to a text file. Thus, when used in conjunction with the IncludeAllProperties parameter, this parameter allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

SizeLimit

Set the maximum number of items to be returned by the cmdlet. Normally, the default size limit is 1000. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

Type

Specify the type of directory objects to find. The cmdlet searches for objects that have one of the 'objectClass' attribute values set to the Type parameter value.

UseDefaultExcludedProperties

When set to 'true', this parameter causes the cmdlet not to load a certain pre-defined set of attributes from the directory to the local memory cache. This pre-defined set of attributes (referred to as "default excluded properties") can be viewed or modified by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Normally, this parameter is used in conjunction with IncudeAllProperties to avoid retrieval of unnecessary data from the directory server, thereby increasing performance of the search operation performed by the cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

UseDefaultExcludedPropertiesExcept

This parameter is deprecated, and has no effect.

WildcardMode

Specify either 'PowerShell' or 'LDAP' as the parameter value. Normally, if this parameter is not supplied, the cmdlet assumes that WildcardMode is set to 'LDAP'. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. The 'PowerShell' value causes the cmdlet to use PowerShell wildcards and quoting rules. Wildcards are processed on the client side, which may result in slow search performance.

For information about PowerShell wildcards and quoting rules, type the following commands at the PowerShell command-prompt:

```
help about_wildcard
help about_quoting_rule
```

The 'LDAP' value causes the cmdlet to use LDAP wildcards (asterisks only) and LDAP quoting rules (backslash as the escape character). Wildcards are processed on the server side, which enables faster search results.

Detailed Description

Use this cmdlet to retrieve users or groups to which a particular Password Settings object is applied. The cmdlet searches the 'msDS-PSOAppliesTo' attribute of the Password Settings object (supplied through the Identity parameter) for objects that meet the search conditions specified.

The output of the cmdlet is a collection of objects, with each object representing one of the directory objects found by the cmdlet. You can pipe the output into another cmdlet, such as Set-QADObject, to make changes to the directory objects found by this cmdlet.

Examples

Example 1

Retrieve all groups affected by the Password Settings object named MyPSO (the groups to which the Password Settings object applies):

```
C:\PS> Get-QADPasswordSettingsObjectAppliesTo 'MyPSO' -Type 'Group'
```

Example 2

Retrieve users and groups affected by the MyPSO Password Settings object that contains the word Administration somewhere in the Description attribute:

```
C:\PS> Get-QADPasswordSettingsObjectAppliesTo 'MyPSO' -Description '*Administration*'
```

Add-QADPasswordSettingsObjectAppliesTo

Add PSO links on a Password Settings object. Active Directory version of Windows Server 2008 or later is required.

Syntax

```
Add-QADPasswordSettingsObjectAppliesTo [-Identity] <IdentityParameter> [-AppliesTo]
<IdentityParameter[]> [-Proxy] [-UseGlobalCatalog] [-Service <String>]
[-ConnectionAccount <String>] [-ConnectionPassword <SecureString>] [-Credential
<PSCredential>] [-Connection <ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

AppliesTo

Specify a list of users or groups to which you want the Password Settings object to apply. Each list entry is the DN, SID, GUID, UPN or Domain\Name of a user or group. Separate the list entries by commas.

Confirm

Prompts you for confirmation before executing the command.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to apply a Password Settings object to users or global security groups. You can specify a list of users and groups, separating the list entries by commas. The cmdlet adds (appends) the specified distinguished names of the users or groups to the 'msDS-PSOAppliesTo' attribute of the Password Settings object, without removing the names that already exist in the attribute.

Examples

Example 1

Apply the Password Settings object to the user object, and display operation results:

```
C:\PS> Add-QADPasswordSettingsObjectAppliesTo 'myPso1' -AppliesTo 'JSmith' | Format-List
```

Example 2

Find a Password Settings object by name and add a PSO link that points to a particular user object (so the Password Settings object applies to that user):

```
C:\PS> Get-QADPasswordSettingsObject -Name 'myPso1' |
Add-QADPasswordSettingsObjectAppliesTo -AppliesTo 'JSmith'
```

Remove-QADPasswordSettingsObjectAppliesTo

Remove PSO links on a Password Settings object. Active Directory version of Windows Server 2008 or later is required.

Syntax

```
Remove-QADPasswordSettingsObjectAppliesTo [-Identity] <IdentityParameter> [-AppliesTo]
<IdentityParameter[]> [-Proxy] [-UseGlobalCatalog] [-Service <String>]
[-ConnectionAccount <String>] [-ConnectionPassword <SecureString>] [-Credential
<PSCredential>] [-Connection <ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

AppliesTo

Specify a list of users or groups that you want the Password Settings object to no longer apply to. Each list entry is the DN, SID, GUID, UPN or Domain\Name of a user or group. Separate the list entries by commas.

Confirm

Prompts you for confirmation before executing the command.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to remove PSO links on a Password Settings object in order to have the Password Settings object no longer apply to certain users or groups. You can specify a list of users and groups, separating the list entries by commas. The cmdlet removes the specified distinguished names of the users or groups from the 'msDS-PSOAppliesTo' attribute of the Password Settings object.

Examples

Example 1

Find a Password Settings object by name, remove a PSO link that points to the group (so the Password Settings object no longer applies to that group), and display operation results:

```
C:\PS> Get-QADPasswordSettingsObject -Name 'myPso1' |
Remove-QADPasswordSettingsObjectAppliesTo -AppliesTo 'myDomain\Account Operators' |
Format-List
```

Get-QADPermission

Retrieve access control entries (ACEs) that meet the conditions you want. Every object returned by this cmdlet represents an access control entry (ACE) in the discretionary access control list (DACL) of a certain directory object.

Syntax

```
Get-QADPermission [-Identity] <IdentityParameter> [-Inherited] [-SchemaDefault]
[-UseTokenGroups] [-UseExtendedMatch] [-Allow] [-Deny] [-ApplyTo
<ArsSecurityInheritance[]>] [-Rights <ActiveDirectoryRights>] [-Account
<IdentityParameter[]>] [-Property <String[]>] [-PropertySet <String[]>] [-ExtendedRight
<String[]>] [-ValidatedWrite <String[]>] [-ChildType <String[]>] [-ApplyToType
<String[]>] [-Proxy] [-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount
<String>] [-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Account

Supply the identity (such as name, Distinguished Name, Domain\Name, SID, etc.) of a security principal (user, group, computer account, etc.). The cmdlet will retrieve ACEs that determine access rights of that security principal on the directory object specified. You can supply identities of multiple accounts.

Allow

Retrieve ACEs that allow access to the directory object specified.

ApplyTo

Depending on the cmdlet you use, this parameter lets you either retrieve or add ACEs that have a particular inheritance type set. Valid parameter values are:

- 'ThisObjectOnly' - Indicates no inheritance. The ACE information is only used on the object on which the ACE is set. ACE information is not inherited by any descendents of the object.
- 'All' - Indicates inheritance that includes the object on which the ACE is set, the object's immediate children, and the descendents of the object's children.
- 'ChildObjects' - Indicates inheritance that includes the object's immediate children and the descendants of the object's children, but not the object itself.

- 'ThisObjectAndImmediateChildObjects' - Indicates inheritance that includes the object itself and its immediate children. It does not include the descendents of its children.
- 'ImmediateChildObjectsOnly' - Indicates inheritance that includes the object's immediate children only, not the object itself or the descendents of its children.

ApplyToType

Retrieve ACEs that can be inherited by objects of a particular class. Property value is the LDAP display name of the classSchema object for the object class you want. (This parameter causes the cmdlet to search by the InheritedObjectType setting on the ACEs.)

You can specify multiple classes, separating the name of the classes by commas. If you do so, the cmdlet retrieves ACEs that can be inherited by objects of any of the classes specified.

ChildType

Retrieve ACEs that control the right to create or delete child objects of a particular class. Parameter value is the LDAP display name of the classSchema object for the child object's class. (This parameter causes the cmdlet to search by the ObjectType setting on the ACEs.)

You can specify multiple classes, separating the names of the classes by commas. If you do so, the cmdlet retrieves ACEs that control the right to create or delete child objects of any of the classes specified.

Deny

Retrieve ACEs that deny access to the directory object specified.

ExtendedRight

Retrieve ACEs that determine the specified extended rights on the directory object. Parameter value is a string array of the names of the extended rights you want. For a list of possible extended rights, see topic "Extended Rights" in the MSDN Library at http://msdn.microsoft.com. For more information about extended rights, see topic "Control Access Rights" in the MSDN Library.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to find.

The cmdlet attempts to retrieve the object identified by this parameter value, disregarding the other search parameters. If you want other search parameters to have effect, do not supply this parameter.

Inherited

Retrieve ACEs that come from security descriptors of the ancestors of the directory object (ACEs that are inherited from the parent container object).

Property

Retrieve ACEs that determine access to the specified attributes of the directory object. Parameter value is a string array of the LDAP display names of the attributes you want.

PropertySet

Retrieve ACEs that determine access to the specified property sets of the directory object. Specify the names of the property sets you want, separating names by commas. For a list of possible property sets, see topic "Property Sets" in the MSDN Library at http://msdn.microsoft.com.

Rights

Depending on the cmdlet you use, this parameter lets you either retrieve or add ACEs that have particular access rights set. Valid parameter values are as follows (for descriptions of these access rights see topic "ActiveDirectoryRights Enumeration" in the MSDN Library at http://msdn.microsoft.com):

- 'ReadProperty'
- 'WriteProperty'
- 'GenericRead'
- 'GenericWrite'
- 'GenericExecute'
- 'GenericAll'
- 'CreateChild'
- 'DeleteChild'
- 'DeleteTree'
- 'ReadControl'
- 'WriteDacl'
- 'WriteOwner'
- 'Synchronize'
- 'AccessSystemSecurity'
- 'ListChildren'
- 'ListObject'
- 'ExtendedRight'
- 'Self'

Parameter value can be any combination of the listed values, separated by commas. For example, the parameter value of 'ReadProperty,WriteProperty' allows you to retrieve or add ACEs that have both the ReadProperty and WriteProperty access rights set.

SchemaDefault

Retrieve ACEs that came from the default security descriptor defined in the classSchema object for the directory object's class.

UseExtendedMatch

Retrieve not only ACEs with the specified access rights setting but also ACEs with other access rights settings that effectively give the same level of access as the rights setting specified.

For example, the -Rights 'ReadProperty' parameter alone causes the cmdlet to retrieve only ACEs that have the ReadProperty access right set, whereas the combination of parameters such as -Rights 'ReadProperty' -UseExtendedMatch also retrieves ACEs that have the GenericRead or GenericAll access right set.

UseTokenGroups

Retrieve ACEs that apply not only to the specified security principal (SID) itself but also to any of the groups to which the account belongs whether directly or because of group nesting.

ValidatedWrite

Retrieve ACEs that determine the specified validated writes on the directory object. Parameter value is a string array of the names of the validated writes you want. For a list of possible validated writes, see topic "Validated Writes" in the MSDN Library at http://msdn.microsoft.com.

Detailed Description

Use this cmdlet to retrieve access control entries (ACEs) from the discretionary access control list (DACL) of a particular object or objects in the directory (directory objects).

The directory objects can be specified using the *Identity* parameter. Another option is to use pipelining: pass the output of the appropriate Get-QAD cmdlet to this cmdlet, with the *-SecurityMask Dacl* parameter supplied for the Get- cmdlet.

The cmdlet returns the objects representing the ACEs that meet the conditions you define using parameters of the cmdlet. You can use pipelining to pass the objects returned by this cmdlet to another cmdlet. For example, you can pass them to the Remove-QADPermission cmdlet in order to delete the respective ACEs from the DACL.

Examples

Example 1

Retrieve the ACEs that are explicitly set on a particular object (the ACEs that are neither inherited from the parent container nor received from the default security descriptor of the respective classSchema object):

```
C:\PS> Get-QADObject 'DistinguishedNameOfTheObject' -SecurityMask Dacl |
Get-QADPermission
```

Example 2

Retrieve all ACEs from the DACL of a particular object (including the ACEs that are inherited from the parent container or received from the default security descriptor of the respective classSchema object):

```
C:\PS> Get-QADObject 'DistinguishedNameOfTheObject' -SecurityMask Dacl |
Get-QADPermission -Inherited -SchemaDefault
```

Example 3

Retrieve the ACEs on a particular object that have any of the specified groups set as the trustee:

```
C:\PS> Get-QADObject 'DistinguishedNameOfTheObject' -SecurityMask Dacl |
Get-QADPermission -Account ('domainName\groupName1','domainName\groupName2')
```

Example 4

Retrieve the ACEs on a particular object that have the trustee set either to the specified user account or to any of the groups to which the user account belongs (whether directly or because of group nesting):

```
C:\PS> Get-QADObject 'DistinguishedNameOfTheObject' -SecurityMask Dacl |
Get-QADPermission -Account 'domain\user' -UseTokenGroups
```

Example 5

Retrieve the ACEs on a particular object that determine Read access to properties of the object:

```
C:\PS> Get-QADObject 'DistinguishedNameOfTheObject' -SecurityMask Dacl |
Get-QADPermission -Rights 'ReadProperty'
```

Example 6

Retrieve the ACEs on a particular user account that are configured with the WriteProperty access right for the 'sAMAccountName' or 'name' property:

```
C:\PS> Get-QADUser 'domain\user' -SecurityMask Dacl | Get-QADPermission -Rights
'WriteProperty' -Property ('sAMAccountName','name')
```

Example 7

Retrieve all the ACEs that allow write access to the 'sAMAccountName' or 'name' property of a particular user account:

```
C:\PS> Get-QADUser 'domain\user' -SecurityMask Dacl | Get-QADPermission -Rights
'WriteProperty' -UseExtendedMatch -Inherited -SchemaDefault -Allow -Property
('sAMAccountName','name')
```

Example 8

Copy the ACEs that are configured on a particular directory object (not including the inherited ACEs or the schema default ACEs) to another directory object:

```
C:\PS> Get-QADPermission 'DistinguishedNameOfSourceObject' | Add-QADPermission
'DistinguishedNameOfDestinationObject'
```

Example 9

Delete all the deny-type ACEs that are configured on a particular directory object (not including the inherited ACEs or the schema default ACEs):

```
C:\PS> Get-QADPermission 'DistinguishedNameOfObject' -Deny | Remove-QADPermission
```

Add-QADPermission

Add access control entries (ACEs) to the discretionary access control list (DACL) of a certain directory object or objects.

Syntax

```
Add-QADPermission [-Identity] <IdentityParameter> -Account <IdentityParameter[]>
[-Rights <ActiveDirectoryRights>] [-Deny] [-ApplyTo <ArsSecurityInheritance>] [-Property
<String[]>] [-PropertySet <String[]>] [-ExtendedRight <String[]>] [-ValidatedWrite
<String[]>] [-ChildType <String[]>] [-ApplyToType <String[]>] [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>] [-WhatIf] [-Confirm]

Add-QADPermission [-Identity] <IdentityParameter> -InputPermission <ArsPermission>
[-Proxy] [-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Account

Supply the identity (such as name, Distinguished Name, Domain\Name, SID, etc.) of a security principal (user, group, computer account, etc.). The cmdlet will add ACEs that determine access rights of that security principal on the directory object specified. You can supply identities of multiple accounts.

ApplyTo

Depending on the cmdlet you use, this parameter lets you either retrieve or add ACEs that have a particular inheritance type set. Valid parameter values are:

- 'ThisObjectOnly' - Indicates no inheritance. The ACE information is only used on the object on which the ACE is set. ACE information is not inherited by any descendents of the object.
- 'All' - Indicates inheritance that includes the object on which the ACE is set, the object's immediate children, and the descendents of the object's children.
- 'ChildObjects' - Indicates inheritance that includes the object's immediate children and the descendants of the object's children, but not the object itself.

- 'ThisObjectAndImmediateChildObjects' - Indicates inheritance that includes the object itself and its immediate children. It does not include the descendents of its children.
- 'ImmediateChildObjectsOnly' - Indicates inheritance that includes the object's immediate children only, not the object itself or the descendents of its children.

ApplyToType

Supply this parameter for the cmdlet to add ACEs that can be inherited by objects of specific classes (rather than all classes). Parameter value is a string array of LDAP display names, each of which identifies the classSchema object for the object class you want.

If you do not supply this parameter, the cmdlet adds ACEs that can be inherited by objects of any class. (This parameter causes the cmdlet to configure the InheritedObjectType setting on the ACEs.)

ChildType

Supply this parameter for the cmdlet to add ACEs that control the right to create or delete child objects of particular classes (rather than all classes). Parameter value is a string array of LDAP display names, each of which identifies the classSchema object for a child object's class you want.

If you do not supply this parameter, the cmdlet configures ACEs to control the right to create or delete child objects of any class. (This parameter causes the cmdlet to configure the ObjectType setting on the ACEs.)

Confirm

Prompts you for confirmation before executing the command.

Deny

Supply this parameter for the cmdlet to add ACEs that deny (rather than allow) access. If you do not supply this parameter, the cmdlet configures ACEs to allow access.

ExtendedRight

Supply this parameter for the cmdlet to add ACEs that determine specific extended rights on the directory object. Parameter value is a string array of the names of the extended rights you want. For a list of possible extended rights, see topic "Extended Rights" in the MSDN Library at http://msdn.microsoft.com. For more information about extended rights, see topic "Control Access Rights" in the MSDN Library.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

InputPermission

This parameter is used to identify the object or objects representing ACEs to add. The parameter accepts parameter values from the pipeline. Thus, when you use pipelining to pass to this cmdlet the objects returned by the Get-QADPermission cmdlet, you should not supply this parameter on the command line. Another option is to save the object in a variable and then supply that variable as a parameter value.

Property

Supply this parameter for the cmdlet to add ACEs that determine access to specific attributes of the directory object (rather than all attributes). Parameter value is a string array of the LDAP display names of the attributes you want.

If you supply neither this parameter nor the PropertySet parameter, the cmdlet configures ACEs to determine access to all attributes.

PropertySet

Supply this parameter if want the cmdlet to add ACEs that determine access to specific property sets of the directory object (rather than all attributes). Parameter value is a string array of the names of the property sets you want. For a list of possible property sets, see topic "Property Sets" in the MSDN Library at http://msdn.microsoft.com.

If you supply neither this parameter nor the Property parameter, the cmdlet configures ACEs to determine access to all attributes.

Rights

Depending on the cmdlet you use, this parameter lets you either retrieve or add ACEs that have particular access rights set. Valid parameter values are as follows (for descriptions of these access rights see topic "ActiveDirectoryRights Enumeration" in the MSDN Library at http://msdn.microsoft.com):

- 'ReadProperty'
- 'WriteProperty'
- 'GenericRead'
- 'GenericWrite'
- 'GenericExecute'
- 'GenericAll'
- 'CreateChild'
- 'DeleteChild'
- 'DeleteTree'
- 'ReadControl'
- 'WriteDacl'
- 'WriteOwner'
- 'Synchronize'
- 'AccessSystemSecurity'
- 'ListChildren'
- 'ListObject'
- 'ExtendedRight'
- 'Self'

Parameter value can be any combination of the listed values, separated by commas. For example, the parameter value of 'ReadProperty,WriteProperty' allows you to retrieve or add ACEs that have both the ReadProperty and WriteProperty access rights set.

ValidatedWrite

Supply this parameter for the cmdlet to add ACEs that determine validated writes on the directory object. Parameter value is a string array of the names of the validated writes you want. For a list of possible validated writes, see topic "Validated Writes" in the MSDN Library at http://msdn.microsoft.com.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to add access control entries (ACEs) to the discretionary access control list (DACL) of a particular object or objects in the directory (directory objects).

The directory objects can be specified using the *Identity* parameter. Another option is to use pipelining: pass the output of the appropriate Get-QAD cmdlet to this cmdlet, with the *-SecurityMask Dacl* parameter supplied for the Get- cmdlet (see examples).

The objects representing ACEs to add can be either passed to this cmdlet through the pipeline or created by the cmdlet itself. In the latter case you use cmdlet parameters to configure the ACEs that you want the cmdlet to add. If you opt to use pipelining, you can have Get-QADPermission retrieve ACEs and then pass the output of that cmdlet to the Add-QADPermission cmdlet so as to copy certain ACEs from one directory object to another directory object (see examples).

Examples

Example 1

Give a certain group full access to a certain organizational unit (OU) and all objects in that OU:

```
C:\PS> Add-QADPermission 'DistinguishedNameOfTheOU' -Account 'domainName\groupName'
-Rights 'GenericAll'
```

Example 2

Deny a certain group permission to modify the sAMAccountName property as well as the properties that are part of the General Information or Web Information property set on a certain user account:

```
C:\PS> Add-QADPermission 'domainName\userName' -Deny -Account 'domainName\groupName'
-Rights 'WriteProperty' -PropertySet ('General-Information','Web-Information')
-Property 'sAMAccountName' -ApplyTo 'ThisObjectOnly'
```

Example 3

Authorize a given group to create user accounts in a particular organizational unit (OU) or in organizational units that are (immediate) children of that OU:

```
C:\PS> Add-QADPermission 'DistinguishedNameOfTheOU' -Account 'domainName\groupName'
-Rights 'CreateChild' -ChildType 'user' -ApplyTo 'ThisObjectAndImmediateChildObjects'
-ApplyToType 'organizationalUnit'
```

Example 4

Authorize a given group to view or modify the group membership list of any group in a particular organizational unit (OU):

```
C:\PS> Add-QADPermission 'DistinguishedNameOfTheOU' -Account 'domainName\groupName'
-Rights 'ReadProperty,WriteProperty' -Property 'member' -ApplyToType 'group'
```

Example 5

Deny a given user account permission to modify the group membership list of any group in a particular organizational unit (OU):

```
C:\PS> Get-QADGroup -SearchRoot 'DistinguishedNameOfTheOU' -SecurityMask 'Dacl' |
Add-QADPermission -Account 'domainName\UserName' -Deny -Rights 'WriteProperty' -Property
'member'
```

Example 6

Authorize a given group to view or modify any property that is part of the Personal Information property set on any user account in a particular organizational unit (OU):

```
C:\PS> Add-QADPermission 'DistinguishedNameOfTheOU' -Account 'domainName\groupName'
–Rights 'ReadProperty,WriteProperty' –PropertySet 'Personal-Information' –ApplyTo
'ChildObjects' –ApplyToType 'user'
```

Example 7

Copy the ACEs that are configured on a particular directory object (not including the inherited or schema default ACEs) to another directory object:

```
C:\PS> Get-QADPermission 'DistinguishedNameOfSourceObject' | Add-QADPermission
'DistinguishedNameOfDestinationObject'
```

Remove-QADPermission

Delete access control entries (ACEs) from the discretionary access control list (DACL) of a directory object or objects.

Syntax

```
Remove-QADPermission [-InputPermission] <ArsPermission> [-Proxy] [-UseGlobalCatalog]
[-Service <String>] [-ConnectionAccount <String>] [-ConnectionPassword <SecureString>]
[-Credential <PSCredential>] [-Connection <ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Confirm

Prompts you for confirmation before executing the command.

InputPermission

This parameter is used to identify the object or objects representing the ACEs to delete. The parameter accepts parameter values from the pipeline. Thus, when you use pipelining to pass to this cmdlet the objects returned by the Get-QADPermission cmdlet, you should not supply this parameter on the command line (see examples). Another option is to save the object in a variable and then supply that variable as a parameter value.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to delete access control entries (ACEs) from the discretionary access control list (DACL) of an object or objects in the directory (directory objects).

The objects representing ACEs to remove can be passed to this cmdlet through the pipeline. You can have Get-QADPermission retrieve ACEs and then pass the output of that cmdlet to the Add-QADPermission cmdlet so as to delete ACEs from the directory object or objects from which the ACEs have been retrieved (see examples).

Examples

Example 1

Delete all the deny-type ACEs that are configured on a particular directory object (not including the inherited ACEs or the schema default ACEs):

```
C:\PS> Get-QADPermission 'DistinguishedNameOfObject' -Deny | Remove-QADPermission
```

Get-QADObjectSecurity

Retrieve security information, such as the owner information or the security descriptor in a string format, from a directory object or objects.

Syntax

```
Get-QADObjectSecurity [-Identity] <IdentityParameter> -Owner [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>]

Get-QADObjectSecurity [-Identity] <IdentityParameter> -Sddl [-Proxy] [-UseGlobalCatalog]
[-Service <String>] [-ConnectionAccount <String>] [-ConnectionPassword <SecureString>]
[-Credential <PSCredential>] [-Connection <ArsConnection>]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

Owner

Supply this parameter for the cmdlet to return an object that represents the owner of the directory object specified by the Identity parameter.

Sddl

Supply this parameter for the cmdlet to return the directory object's security descriptor in a string format.

The string format is defined by the security descriptor definition language (SDDL). You can use the string format to store or transmit the security descriptor. For a description of the string format, see topic "Security Descriptor Definition Language" in the MSDN Library at http://msdn.microsoft.com.

Detailed Description

Use this cmdlet to retrieve security information from an object in the directory (directory object). Thus, you can get an object representing the owner of a particular directory object. You can also have this cmdlet return the security descriptor of a directory object in a string format defined by the security descriptor definition language (SDDL).

Examples

Example 1

Get the object that represents the owner of a particular group:

```
C:\PS> Get-QADObjectSecurity 'domainName\groupName' -Owner
```

Example 2

For a particular directory object, list the security descriptor in a string format:

```
C:\PS> Get-QADObjectSecurity 'DistinguishedNameOfTheObject' -SDDL
```

Example 3

For every computer object held in the Computers container in domain dom.lab.local, list the distinguished name of the owner of the computer object:

```
C:\PS> Get-QADComputer -SearchRoot 'dom.lab.local/Computers' -SecurityMask 'Owner' |
ForEach-Object {$computer=$_; Get-QADObjectSecurity $_ -Owner | Select-Object
@{Name='Computer'; Expression={$computer.DN}}, @{Name='Owner'; Expression={$_.DN}}}
```

Set-QADObjectSecurity

Update security information on a directory object or objects. You can change the owner of an object or change the option that governs protection of an object from the effects of inherited rights.

Syntax

```
Set-QADObjectSecurity [-Identity] <IdentityParameter> -LockInheritance [-Remove]
[-Proxy] [-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>] [-WhatIf] [-Confirm]

Set-QADObjectSecurity [-Identity] <IdentityParameter> -Owner <IdentityParameter>
[-Proxy] [-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>] [-WhatIf] [-Confirm]

Set-QADObjectSecurity [-Identity] <IdentityParameter> -UnlockInheritance [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Parameters

Confirm

Prompts you for confirmation before executing the command.

Identity

Specify the Distinguished Name (DN), Canonical Name, GUID or, if applicable, the Domain\Name, UPN or SID of the object you want the cmdlet to act upon. The cmdlet makes changes to, or otherwise manages, the object identified by this parameter value. When you pipe an object into the cmdlet, this parameter is used to receive the object.

LockInheritance

Supply this parameter for the cmdlet to configure the security descriptor on the directory object so that access control entries (ACEs) that are set on the discretionary access control list (DACL) of the parent container, and any objects above the parent container in the directory hierarchy, are not applied to the DACL of that directory object.

Owner

Specify the identity (name, Distinguished Name, Domain\Name, SID, etc.) of the security principal that you want the cmdlet to set as the owner of the directory object specified by the Identity parameter. Another option is to get an object representing the owner you want, save the object in a variable, and supply that variable as a value for this parameter.

Remove

This parameter can be used in conjunction with the LockInheritance parameter to remove the inherited ACEs from the directory object.

If you supply this parameter, the cmdlet removes the ACEs that were previously applied (inherited) from the parent and keeps only those ACEs that are explicitly defined on the directory object.

If you do not supply this parameter, the cmdlet copies the ACEs that were previously applied from the parent, merging them with the ACEs that are explicitly defined on the directory object.

UnlockInheritance

Supply this parameter for the cmdlet to configure the security descriptor on the directory object so that access control entries (ACEs) originating from the parent container are applied to the DACL of that directory object in accord with the inheritance flags set on those ACEs.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

You can use this cmdlet to perform any of the following tasks on a particular directory object (each of these tasks implies certain changes to the security descriptor of the directory object):

- Set a given security principal to be the owner of that object.
- Specify whether access control entries (ACEs) that are set on the discretionary access control list (DACL) of the parent container, and any objects above the parent container in the directory hierarchy, are applied to the object's DACL.

With the latter task, consider that ACEs can be set on a container object, such as an organizationalUnit, domainDNS, container, and so on, and propagated to child objects based on the inheritance flags set on those ACEs. If you want to explicitly control the ACEs on a certain sensitive object, such as a private OU or a special user, you can prevent ACEs from being propagated to the object by its parent container or its parent container's predecessors.

Examples

Example 1

For a given directory object, set a certain group as the owner of the object:

```
C:\PS> Set-QADObjectSecurity 'DistinguishedNameOfTheObject' -Owner
'domainName\userName'
```

Example 2

Prevent a certain user account from inheriting ACEs from the parent object and remove the ACEs that were previously applied from the parent object or its ancestors. As a result, access to the user account is controlled by only those ACEs that are explicitly set on the account:

```
C:\PS> Set-QADObjectSecurity 'domainName\userName' -LockInheritance -Remove
```

Example 3

Configure security settings on a particular user account to allow inheritable ACEs from the parent container to propagate to that user account, merging them with those ACEs that are explicitly set on the user account:

```
C:\PS> Set-QADObjectSecurity 'domain\user' -UnlockInheritance
```

Example 4

For every computer object held in a given organizational unit (OU), set the owner of the computer object to the Administrators domain local group:

```
C:\PS> Get-QADComputer -SearchRoot 'DistinguishedNameOfTheOU' –SecurityMask 'Owner' |
Set-QADObjectSecurity –Owner 'domainName\administrators'
```

Get-QADRootDSE

Retrieve the rootDSE object from the current directory server (domain controller).

Syntax

```
Get-QADRootDSE [-Proxy] [-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount
<String>] [-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>]
```

Parameters

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Detailed Description

This cmdlet returns the rootDSE object containing data about the directory server. The rootDSE object is retrieved from a domain controller that is specific to the current connection. Thus, if connection parameters are supplied to choose a certain domain controller, the cmdlet retrieves the rootDSE object from that domain controller.

You can use the rootDSE object to get distinguished names of the domain, schema, and configuration containers, and other data about the server and the contents of its directory data tree. For information about attributes supported by rootDSE, refer to the "RootDSE" topic in the Active Directory Schema documentation in the MSDN Library (http://msdn.microsoft.com).

When connected to ActiveRoles Server, the cmdlet retrieves the rootDSE object containing information about the ActiveRoles Server namespaces. For information about attributes supported by the ActiveRoles Server rootDSE, refer to the ActiveRoles Server SDK and Resource Kit documentation (this documentation is normally installed with the ActiveRoles Server Administration Service).

Examples

Example 1

List the distinguished names of the domain, schema, and configuration containers for the current connection (this command retrieves and displays the values of the following attributes from rootDSE: defaultNamingContext, schemaNamingContext, and configurationNamingContext):

```
C:\PS> get-QADRootDSE | Format-List {$_["defaultNamingContext"]},
{$_["schemaNamingContext"]}, {$_["configurationNamingContext"]}
```

Example 2

Identify the domain controller that is used by the current connection. The output of this command is the distinguished name of the server object for that domain controller in the configuration container (the command displays the value of the serverName attribute retrieved from rootDSE):

```
C:\PS> (get-QADRootDSE)["serverName"]
```

Example 3

Connect to any available ActiveRoles Server Administration Service and then retrieve the fully qualified domain name of the computer running the Administration Service to which you have connected:

```
C:\PS> connect-QADService -proxy
C:\PS> (get-QADRootDSE)["edsvaServiceFullDns"]
```

Cmdlet Reference - ActiveRoles Server

Here you can find information about command-line tools (cmdlets) that are provided by ActiveRoles Management Shell.

This section covers the cmdlets for managing configuration data and other data specific to ActiveRoles Server.

All cmdlets of this category require a connection to be established to the ActiveRoles Server Administration Service by supplying the *Proxy* parameter.

Get-QARSAccessTemplate

Retrieve Access Template objects from ActiveRoles Server.

Syntax

```
Get-QARSAccessTemplate [[-Identity] <IdentityParameter>] [-Predefined] [-SearchRoot
<IdentityParameter>] [-SearchScope <SearchScope>] [-AttributeScopeQuery <String>]
[-PageSize <Int32>] [-SizeLimit <Int32>] [-LdapFilter <String>] [-WildcardMode
<WildcardMode>] [-ObjectAttributes <Object>] [-Description <String>] [-DisplayName
<String>] [-Name <String>] [-Anr <String>] [-CreatedOn <DateTime>] [-CreatedAfter
<DateTime>] [-CreatedBefore <DateTime>] [-LastChangedOn <DateTime>] [-LastChangedAfter
<DateTime>] [-LastChangedBefore <DateTime>] [-IncludeAllProperties]
[-DontConvertValuesToFriendlyRepresentation] [-SerializeValues]
[-ReturnPropertyNamesOnly] [-DontUseDefaultIncludedProperties]
[-UseDefaultExcludedProperties] [-ExcludedProperties <String[]>] [-IncludedProperties
<String[]>] [-UseDefaultExcludedPropertiesExcept <String[]>] [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Note that this cmdlet requires a connection to the ActiveRoles Server Administration Service, so the *Proxy* parameter must be used to establish a connection.

Parameters

Anr

Specify a value to be resolved using ambiguous name resolution (ANR). Which attributes are included in an ANR search depends upon the Active Directory schema. Thus, in Windows Server 2003 based Active Directory, the following attributes are set for ANR by default:

- Display-Name (displayName)
- Given-Name (givenName)
- Legacy-Exchange-DN (legacyExchangeDN)
- ms-DS-Additional-Sam-Account-Name (msDS-AdditionalSamAccountName)
- Physical-Delivery-Office-Name (physicalDeliveryOfficeName)
- Proxy-Addresses (proxyAddresses)
- RDN (name)

- SAM-Account-Name (sAMAccountName)
- Surname (sn)

For instance, when you supply 'ann*' as the value of this parameter, the cmdlet searches for objects that have ann at the beginning of the value of at least one of the attributes listed above.

AttributeScopeQuery

Specify the LDAP display name of an attribute that has DN syntax (for example, "member" or "memberOf"). The cmdlet enumerates the Distinguished Name values of the attribute on the object specified by the SearchRoot parameter, and performs the search on the objects represented by the Distinguished Names. The SearchScope parameter has no effect in this case. The object to search must be specified by using the SearchRoot parameter rather than the Identity parameter.

For instance, with the value of this parameter set to "memberOf", the cmdlet searches the collection of the groups to which the SearchRoot object belongs.

CreatedAfter

Specify the lower boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created after the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedBefore

Specify the upper boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created before the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedOn

Specify the object creation date by which to filter objects found, searching for objects created within the date specified. This parameter is mutually exclusive with the CreatedAfter and CreatedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

Description

Search by the 'description' attribute.

DisplayName

Search by the 'displayName' attribute.

DontConvertValuesToFriendlyRepresentation

This parameter causes the cmdlet to represent the Integer8 and OctetString attribute values “as is,” without converting them to a user-friendly, human-readable form. If this parameter is omitted, the cmdlet performs the following data conversions:

- The values of the Integer8 attributes listed in the Integer8AttributesThatContainDateTimes array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to DateTime.

- The values of the Integer8 attributes listed in the Integer8AttributesThatContainNegativeTimeSpans array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to TimeSpan.
- The values of the other Integer8 attributes are converted from IADsLargeInteger to Int64.
- The values of the OctetString attributes are converted from byte[] to BinHex strings.

Note: This parameter has an effect only on the properties of the output object that have the member type of NoteProperty. Such properties are normally added to the output object in order to provide access to the attribute values of the respective directory object that are loaded to the local memory cache but cannot be accessed by using properties of the base object (the object for which the output object serves as a wrapper).

DontUseDefaultIncludedProperties

This parameter causes the cmdlet to load only a small set of attributes from the directory to the local memory cache (normally, this set is limited to objectClass and ADsPath). Other attributes are retrieved from the directory as needed when you use the cmdlet output objects to read attribute values. Thus, if you want only to count the objects that meet certain conditions (rather than examine values of particular attributes), then you can use this parameter to increase performance of your search. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

ExcludedProperties

Use this parameter to specify the attributes that you do not want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the ExcludedProperties parameter you can change this default behavior on an ad-hoc basis, in order to prevent certain attributes from being loaded. Another scenario involves the use of this parameter in conjunction with IncludeAllProperties in order to restrict the set of the cached attributes.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

Identity

If you want the cmdlet to retrieve a single Access Template, specify the name, Canonical Name, or Distinguished Name of the Access Template as the value of this parameter. If you want to search for Access Templates by other properties (for example, using an LDAP filter), omit this parameter.

IncludeAllProperties

With this parameter, the cmdlet retrieves all attributes of the respective directory object (such as a User object), and stores the attribute values in the memory cache on the local computer. Attribute values can be read from the memory cache by using properties of the object returned by the cmdlet. Thus, when used in conjunction with the SerializeValues parameter, it allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

IncludedProperties

Use this parameter to specify the attributes that you want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the IncludedProperty parameter you can direct the cmdlet to cache some attributes in addition to the default set.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet.

LastChangedAfter

Specify the lower boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed after the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedBefore

Specify the upper boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed before the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedOn

Specify the object modification date by which to filter objects found, searching for objects that have last changed within the date specified. This parameter is mutually exclusive with the LastChangedAfter and LastChangedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

LdapFilter

Specify the LDAP search filter that defines your search criteria. Note that the search filter string is case-sensitive.

The cmdlet disregards this parameter if an Identity value is supplied. If you want this parameter to have effect, do not supply an Identity parameter value.

If you supply the LdapFilter parameter along with attribute-specific parameters, then your search returns objects that meet the conditions defined by the LDAP filter and have the specified attributes set to the specified values.

Name

Search by the 'name' attribute.

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet searches for objects that have the specified attributes set to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to search. A value may include an asterisk character - a wildcard representing any group of characters. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

PageSize

Set the maximum number of items in each page of the search results that will be returned by the cmdlet. After the directory server has found the number of objects that are specified by this parameter, it will stop searching and return the results to the cmdlet. When the cmdlet requests more data, the server will restart the search where it left off. You can use this setting to adjust the number of requests (network calls) to the directory server issued by the cmdlet during a search.

Normally, the default page size is 50. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

Predefined

Set this parameter to 'true' for the cmdlet to retrieve only those Access Templates that are marked "predefined" in ActiveRoles Server. The predefined Access Templates are installed with ActiveRoles Server, and cannot be modified or deleted.

ReturnPropertyNamesOnly

This parameter causes the cmdlet to list the names of the object attributes whose values the cmdlet retrieves from the directory and stores in the memory cache on the local computer. Thus, when used in conjunction with the IncludeAllProperties parameter, it lists the names of all attributes of the respective directory object (such as a User object). For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

SearchRoot

Specify the Distinguished Name, Canonical Name, or GUID of the container to search. By default, the cmdlet searches the entire sub-tree of which SearchRoot is the topmost object (sub-tree search). This default behavior can be altered by using the SearchScope parameter.

The search criteria are defined by the LdapFilter parameter value and the values of attribute-specific parameters.

The cmdlet disregards this parameter if an Identity value is supplied. If you want this parameter to have effect, do not supply any Identity parameter value.

SearchScope

Specify one of these parameter values:

- 'Base' - Limits the search to the base (SearchRoot) object. The result contains a maximum of one object.
- 'OneLevel' - Searches the immediate descendant (child) objects of the base (SearchRoot) object, excluding the base object.
- 'Subtree' - Searches the whole sub-tree, including the base (SearchRoot) object and all its descendant objects.

Normally, if this parameter is not supplied, the cmdlet performs a Subtree search. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

SerializeValues

This parameter causes the cmdlet to output an object whose properties store the attribute values of the respective directory object that are loaded to the local memory cache. The value returned by each property of the output object is represented as a string (serialized) so as to facilitate the export of the attribute values to a text file. Thus, when used in conjunction with the IncludeAllProperties parameter, this parameter allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

SizeLimit

Set the maximum number of items to be returned by the cmdlet. Normally, the default size limit is 1000. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

UseDefaultExcludedProperties

When set to 'true', this parameter causes the cmdlet not to load a certain pre-defined set of attributes from the directory to the local memory cache. This pre-defined set of attributes (referred to as "default excluded properties") can be viewed or modified by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Normally, this parameter is used in conjunction with IncudeAllProperties to avoid retrieval of unnecessary data from the directory server, thereby increasing performance of the search operation performed by the cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

UseDefaultExcludedPropertiesExcept

This parameter is deprecated, and has no effect.

WildcardMode

Specify either 'PowerShell' or 'LDAP' as the parameter value. Normally, if this parameter is not supplied, the cmdlet assumes that WildcardMode is set to 'LDAP'. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. The 'PowerShell' value causes the cmdlet to use PowerShell wildcards and quoting rules. Wildcards are processed on the client side, which may result in slow search performance.

For information about PowerShell wildcards and quoting rules, type the following commands at the PowerShell command-prompt:

```
help about_wildcard
help about_quoting_rule
```

The 'LDAP' value causes the cmdlet to use LDAP wildcards (asterisks only) and LDAP quoting rules (backslash as the escape character). Wildcards are processed on the server side, which enables faster search results.

Detailed Description

Use this cmdlet to retrieve ActiveRoles Server Access Template objects that meet the conditions you specify. Each Access Template object contains information about a certain Access Template. Access Template objects can be used as input to *-QARSAccessTemplateLink cmdlets for managing Access Template links. For background information about Access Templates, see ActiveRoles Server Administrator Guide.

Examples

Example 1

Connect to any available Administration Service and list the names of all pre-defined ActiveRoles Server Access Templates located in a certain container:

```
C:\PS> connect-QADService -Proxy
C:\PS> get-QARSAccessTemplate -SearchRoot 'Configuration/Access Templates/Builtin'
-Predefined $true | format-List Name, ParentContainerDN
```

Example 2

List the general-purpose Access Templates for Active Directory data management that are included with ActiveRoles Server by default:

```
C:\PS> connect-QADService -Proxy
C:\PS> get-QARSAccessTemplate -SearchRoot 'Configuration/Access Templates/Active
Directory' -SearchScope 'OneLevel' -Predefined $true | format-List Name
```

Get-QARSAccessTemplateLink

Retrieve Access Template Link objects from ActiveRoles Server.

Syntax

```
Get-QARSAccessTemplateLink [[-Identity] <IdentityParameter>] [-DirectoryObject
<IdentityParameter[]>] [-Trustee <IdentityParameter[]>] [-AccessTemplate
<IdentityParameter[]>] [-Enabled] [-Disabled] [-AppliedTo <ATLinkFlags>]
[-SynchronizedToAD] [-Predefined] [-SearchRoot <IdentityParameter>] [-SearchScope
<SearchScope>] [-AttributeScopeQuery <String>] [-PageSize <Int32>] [-SizeLimit <Int32>]
[-LdapFilter <String>] [-WildcardMode <WildcardMode>] [-ObjectAttributes <Object>]
[-Description <String>] [-DisplayName <String>] [-Name <String>] [-Anr <String>]
[-CreatedOn <DateTime>] [-CreatedAfter <DateTime>] [-CreatedBefore <DateTime>]
[-LastChangedOn <DateTime>] [-LastChangedAfter <DateTime>] [-LastChangedBefore
<DateTime>] [-IncludeAllProperties] [-DontConvertValuesToFriendlyRepresentation]
[-SerializeValues] [-ReturnPropertyNamesOnly] [-DontUseDefaultIncludedProperties]
[-UseDefaultExcludedProperties] [-ExcludedProperties <String[]>] [-IncludedProperties
<String[]>] [-UseDefaultExcludedPropertiesExcept <String[]>] [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Note that this cmdlet requires a connection to the ActiveRoles Server Administration Service, so the *Proxy* parameter must be used to establish a connection.

Parameters

AccessTemplate

Specify the identity (such as name, Distinguished Name, etc.) of an Access Template for the cmdlet to retrieve Access Template links that apply the Access Template specified.

Anr

Specify a value to be resolved using ambiguous name resolution (ANR). Which attributes are included in an ANR search depends upon the Active Directory schema. Thus, in Windows Server 2003 based Active Directory, the following attributes are set for ANR by default:

- Display-Name (displayName)
- Given-Name (givenName)
- Legacy-Exchange-DN (legacyExchangeDN)
- ms-DS-Additional-Sam-Account-Name (msDS-AdditionalSamAccountName)

- Physical-Delivery-Office-Name (physicalDeliveryOfficeName)
- Proxy-Addresses (proxyAddresses)
- RDN (name)
- SAM-Account-Name (sAMAccountName)
- Surname (sn)

For instance, when you supply 'ann*' as the value of this parameter, the cmdlet searches for objects that have ann at the beginning of the value of at least one of the attributes listed above.

AppliedTo

Depending on the cmdlet you use, this parameter lets you either retrieve or configure Access Template links that have specific settings for permission inheritance. Valid parameter values are:

- 'This' - Indicates no inheritance. The Access Template link information is only used on the object to which the Access Template is applied. Access Template link information is not inherited by any descendents of the object.
- 'ThisObjectAndAllChildObjects' - Indicates inheritance that includes the object to which the Access Template is applied, the object's immediate children, and the descendents of the object's children.
- 'ThisObjectAndImmediateChildObjects' - Indicates inheritance that includes the object itself and its immediate children. It does not include the descendents of its children.
- 'AllChildObjects' - Indicates inheritance that includes the object's immediate children and the descendants of the object's children, but not the object itself.
- 'ImmediateChildObjects' - Indicates inheritance that includes the object's immediate children only, not the object itself or the descendents of its children.

AttributeScopeQuery

Specify the LDAP display name of an attribute that has DN syntax (for example, "member" or "memberOf"). The cmdlet enumerates the Distinguished Name values of the attribute on the object specified by the SearchRoot parameter, and performs the search on the objects represented by the Distinguished Names. The SearchScope parameter has no effect in this case. The object to search must be specified by using the SearchRoot parameter rather than the Identity parameter.

For instance, with the value of this parameter set to "memberOf", the cmdlet searches the collection of the groups to which the SearchRoot object belongs.

CreatedAfter

Specify the lower boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created after the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedBefore

Specify the upper boundary of the object creation date and time by which to filter objects found. The cmdlet returns only the objects that were created before the date and time specified. Supplying both CreatedAfter and CreatedBefore bounds a time interval for the objects' creation. If you supply only CreatedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

CreatedOn

Specify the object creation date by which to filter objects found, searching for objects created within the date specified. This parameter is mutually exclusive with the CreatedAfter and CreatedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

Description

Search by the 'description' attribute.

DirectoryObject

Specify the identity (such as name, Distinguished Name, Domain\Name, etc.) of a directory object for the cmdlet to retrieve Access Template links that determine ActiveRoles Server security settings on that object.

Disabled

Supply this parameter for the cmdlet to retrieve only those Access Template links that are configured to have no effect in ActiveRoles Server (disabled links).

DisplayName

Search by the 'displayName' attribute.

DontConvertValuesToFriendlyRepresentation

This parameter causes the cmdlet to represent the Integer8 and OctetString attribute values "as is," without converting them to a user-friendly, human-readable form. If this parameter is omitted, the cmdlet performs the following data conversions:

- The values of the Integer8 attributes listed in the Integer8AttributesThatContainDateTimes array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to DateTime.
- The values of the Integer8 attributes listed in the Integer8AttributesThatContainNegativeTimeSpans array (see parameter descriptions for the Get- and Set-QADPSSnapinSettings cmdlets) are converted from IADsLargeInteger to TimeSpan.
- The values of the other Integer8 attributes are converted from IADsLargeInteger to Int64.
- The values of the OctetString attributes are converted from byte[] to BinHex strings.

Note: This parameter has an effect only on the properties of the output object that have the member type of NoteProperty. Such properties are normally added to the output object in order to provide access to the attribute values of the respective directory object that are loaded to the local memory cache but cannot be accessed by using properties of the base object (the object for which the output object serves as a wrapper).

DontUseDefaultIncludedProperties

This parameter causes the cmdlet to load only a small set of attributes from the directory to the local memory cache (normally, this set is limited to objectClass and ADsPath). Other attributes are retrieved from the directory as needed when you use the cmdlet output objects to read attribute values. Thus, if you want only to count the objects that meet certain conditions (rather than examine values of particular attributes), then you can use this parameter to increase performance of your search. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

Enabled

Supply this parameter for the cmdlet to retrieve only those Access Template links that are configured to have effect in ActiveRoles Server (enabled links).

ExcludedProperties

Use this parameter to specify the attributes that you do not want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the ExcludedProperties parameter you can change this default behavior on an ad-hoc basis, in order to prevent certain attributes from being loaded. Another scenario involves the use of this parameter in conjunction with IncludeAllProperties in order to restrict the set of the cached attributes.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

Identity

If you want the cmdlet to retrieve a single Access Template Link object by name, specify the name, Canonical Name, or Distinguished Name of the respective AT Link object (located in the 'Configuration/AT Links' container in the ActiveRoles Server Configuration namespace). If you want to search for AT Links by other properties, omit this parameter.

IncludeAllProperties

With this parameter, the cmdlet retrieves all attributes of the respective directory object (such as a User object), and stores the attribute values in the memory cache on the local computer. Attribute values can be read from the memory cache by using properties of the object returned by the cmdlet. Thus, when used in conjunction with the SerializeValues parameter, it allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

IncludedProperties

Use this parameter to specify the attributes that you want the cmdlet to retrieve from the directory and store in the memory cache on the local computer. Supply a list of the attribute LDAP display names as the parameter value. By default, the cmdlet caches a certain pre-defined set of attributes, which you can view or modify by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Using the IncludedProperty parameter you can direct the cmdlet to cache some attributes in addition to the default set.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet.

LastChangedAfter

Specify the lower boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed after the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedAfter, there is no upper boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedBefore

Specify the upper boundary of the object modification date and time by which to filter objects found. The cmdlet returns only the objects that have last changed before the date and time specified. Supplying both LastChangedAfter and LastChangedBefore bounds a time interval for the objects' last change. If you supply only LastChangedBefore, there is no lower boundary on the date. Parameter value is a DateTime object that specifies the date and time you want.

LastChangedOn

Specify the object modification date by which to filter objects found, searching for objects that have last changed within the date specified. This parameter is mutually exclusive with the LastChangedAfter and LastChangedBefore parameters. Parameter value is a DateTime object that specifies the date you want.

LdapFilter

Specify the LDAP search filter that defines your search criteria. Note that the search filter string is case-sensitive.

The cmdlet disregards this parameter if an Identity value is supplied. If you want this parameter to have effect, do not supply an Identity parameter value.

If you supply the LdapFilter parameter along with attribute-specific parameters, then your search returns objects that meet the conditions defined by the LDAP filter and have the specified attributes set to the specified values.

Name

Search by the 'name' attribute.

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet searches for objects that have the specified attributes set to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to search. A value may include an asterisk character - a wildcard representing any group of characters. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

PageSize

Set the maximum number of items in each page of the search results that will be returned by the cmdlet. After the directory server has found the number of objects that are specified by this parameter, it will stop searching and return the results to the cmdlet. When the cmdlet requests more data, the server will restart the search where it left off. You can use this setting to adjust the number of requests (network calls) to the directory server issued by the cmdlet during a search.

Normally, the default page size is 50. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

Predefined

Set this parameter to 'true' for the cmdlet to retrieve only those Access Template links that are marked "predefined" in ActiveRoles Server. The predefined Access Template links are installed with ActiveRoles Server, and cannot be modified or deleted.

ReturnPropertyNamesOnly

This parameter causes the cmdlet to list the names of the object attributes whose values the cmdlet retrieves from the directory and stores in the memory cache on the local computer. Thus, when used in conjunction with the IncludeAllProperties parameter, it lists the names of all attributes of the respective directory object (such as a User object). For examples of how to use this parameter, see documentation on the Get-QADUser or Get-QADObject cmdlet.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

SearchRoot

Specify the Distinguished Name, Canonical Name, or GUID of the container to search. By default, the cmdlet searches the entire sub-tree of which SearchRoot is the topmost object (sub-tree search). This default behavior can be altered by using the SearchScope parameter.

The search criteria are defined by the LdapFilter parameter value and the values of attribute-specific parameters.

The cmdlet disregards this parameter if an Identity value is supplied. If you want this parameter to have effect, do not supply any Identity parameter value.

SearchScope

Specify one of these parameter values:

- 'Base' - Limits the search to the base (SearchRoot) object. The result contains a maximum of one object.
- 'OneLevel' - Searches the immediate descendant (child) objects of the base (SearchRoot) object, excluding the base object.
- 'Subtree' - Searches the whole sub-tree, including the base (SearchRoot) object and all its descendant objects.

Normally, if this parameter is not supplied, the cmdlet performs a Subtree search. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

SerializeValues

This parameter causes the cmdlet to output an object whose properties store the attribute values of the respective directory object that are loaded to the local memory cache. The value returned by each property of the output object is represented as a string (serialized) so as to facilitate the export of the attribute values to a text file. Thus, when used in conjunction with the IncludeAllProperties parameter, this parameter allows an entire object to be exported from the directory to a text file. For examples of how to use this parameter, see documentation on the Get-QADUser cmdlet.

SizeLimit

Set the maximum number of items to be returned by the cmdlet. Normally, the default size limit is 1000. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

SynchronizedToAD

Set this parameter to 'true' for the cmdlet to retrieve only those Access Template links that are configured with the option to propagate permission settings to Active Directory. To retrieve only those links that do not propagate permission settings to Active Directory, set this parameter to 'false'.

Trustee

Specify the identity (such as name, Distinguished Name, Domain\Name, SID, etc.) of a security principal object (such as user or group) for the cmdlet to retrieve Access Template links that determine access rights given to that object in ActiveRoles Server.

UseDefaultExcludedProperties

When set to 'true', this parameter causes the cmdlet not to load a certain pre-defined set of attributes from the directory to the local memory cache. This pre-defined set of attributes (referred to as "default excluded properties") can be viewed or modified by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. Normally, this parameter is used in conjunction with IncudeAllProperties to avoid retrieval of unnecessary data from the directory server, thereby increasing performance of the search operation performed by the cmdlet.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

UseDefaultExcludedPropertiesExcept

This parameter is deprecated, and has no effect.

WildcardMode

Specify either 'PowerShell' or 'LDAP' as the parameter value. Normally, if this parameter is not supplied, the cmdlet assumes that WildcardMode is set to 'LDAP'. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively. The 'PowerShell' value causes the cmdlet to use PowerShell wildcards and quoting rules. Wildcards are processed on the client side, which may result in slow search performance.

For information about PowerShell wildcards and quoting rules, type the following commands at the PowerShell command-prompt:

```
help about_wildcard
help about_quoting_rule
```

The 'LDAP' value causes the cmdlet to use LDAP wildcards (asterisks only) and LDAP quoting rules (backslash as the escape character). Wildcards are processed on the server side, which enables faster search results.

Detailed Description

Use this cmdlet to retrieve ActiveRoles Server Access Template Link objects (also referred to as Access Template links) that meet the conditions you specify. Each Access Template link contains information on how a certain Access Template is applied to determine access rights of a certain security principal (Trustee) on a certain directory object (securable object). Access Template Link objects can be used as input to *-QARSAccessTemplateLink cmdlets for managing Access Template link data. For background information about Access Templates, see ActiveRoles Server Administrator Guide.

Examples

Example 1

Connect to any available Administration Service and, for every Access Template link, list the distinguished names of the following entities:

- **Directory object** The securable object to which the given link applies an Access Template.
- **Access Template** The Access Template that is applied by the given link.
- **Trustee** The security principal whose access rights on the securable object are specified by the given link.

Namely, this command retrieves and displays the values of the DirectoryObject, AccessTemplate, and Trustee properties of the objects returned by the cmdlet:

```
C:\PS> connect-QADService -Proxy
C:\PS> get-QARSAccessTemplateLink | format-List DirectoryObject, AccessTemplate, Trustee
```

Example 2

Given the name of an Access Template, list all objects on which the Access Template determines security settings (for each link that is based on that Access Template, list the securable object to which the link is applied). This command retrieves and displays the value of the DirectoryObject property of the objects returned by the cmdlet:

```
C:\PS> connect-QADService -Proxy
C:\PS> get-QARSAccessTemplateLink -AccessTemplate 'AR Server Security - Active Directory
Container'| format-List DirectoryObject
```

Example 3

Given the name of an Access Template, list all objects that have their access rights defined by using the given Access Template (for each link that is based on that Access Template, list the security principal to which the link points). This command retrieves and displays the value of the Trustee property of the objects returned by the cmdlet:

```
C:\PS> connect-QADService -Proxy
C:\PS> get-QARSAccessTemplateLink -AccessTemplate 'AR Server Security - Active Directory
Container'| format-List Trustee
```

Example 4

Given the pre-Windows 2000 name of a group, list all the Access Templates that determine access rights of that group (find all Access Template links that have the given group set as the security principal, and then, for every such link, list the Access Template on which the link is based and the securable object to which the link is applied):

```
C:\PS> connect-QADService -Proxy
C:\PS> get-QARSAccessTemplateLink –Trustee 'domainName\groupName' | format-List
DirectoryObject, AccessTemplate
```

Example 5

Given the name of an ActiveRoles Server Managed Unit (MU), list all the Access Templates that determine security settings on that MU (find all Access Template links that have the given MU set as the securable object, and then, for every such link, list the Access Template on which the link is based and the security principal to which the link points):

```
C:\PS> connect-QADService -Proxy
C:\PS> get-QARSAccessTemplateLink –DirectoryObject 'Configuration/Managed
Units/ManagedUnitName' | format-List Trustee, AccessTemplate
```

Example 6

For a given organizational unit (OU), list the objects in that OU that have native Active Directory permission settings defined by using any Access Template (find all the Access Templates linked to any object in the OU with the option to synchronize the resulting permission settings to Active Directory):

```
C:\PS> connect-QADService -Proxy
C:\PS> get-QADObject -SearchRoot 'OrganizationalUnitName' | %{get-QARSAccessTemplateLink
-DirectoryObject $_ –SynchronizedToAD $true | format-List AccessTemplate,
DirectoryObject}
```

Example 7

Given the name of an Access Template and the name of an organizational unit (OU), remove all security settings on that OU that are determined by that Access Template (remove all links that are based on the given Access Template and applied to the given OU):

```
C:\PS> connect-QADService -Proxy
C:\PS> get-QARSAccessTemplateLink -AccessTemplate 'AccessTemplateName' -DirectoryObject
'OrganizationalUnitName' -Predefined $false | remove-QARSAccessTemplateLink -Confirm
```

Example 8

Given the name of an Access Template and the pre-Windows 2000 name of a group, revoke all access rights from that group that are defined by using that Access Template (remove all links that are based on the given Access Template and point to the given group):

```
C:\PS> connect-QADService -Proxy
C:\PS> get-QARSAccessTemplateLink -AccessTemplate 'AccessTemplateName' -Trustee
'domainName\groupName' -Predefined $false | remove-QARSAccessTemplateLink -Confirm
```

Set-QARSAccessTemplateLink

Make changes to existing links of ActiveRoles Server Access Templates.

Syntax

```
Set-QARSAccessTemplateLink [-Identity] <IdentityParameter> [-ObjectAttributes <Object>]
[-AccessTemplate <IdentityParameter>] [-Trustee <IdentityParameter>] [-Description
<String>] [-AppliedTo <ATLinkFlags>] [-SynchronizedToAD] [-Disabled] [-Enabled] [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Note that this cmdlet requires a connection to the ActiveRoles Server Administration Service, so the *Proxy* parameter must be used to establish a connection.

Parameters

AccessTemplate

Specify the identity (such as name, Distinguished Name, etc.) of an Access Template you want. The cmdlet configures the link to apply that Access Template.

AppliedTo

Depending on the cmdlet you use, this parameter lets you either retrieve or configure Access Template links that have specific settings for permission inheritance. Valid parameter values are:

- 'This' - Indicates no inheritance. The Access Template link information is only used on the object to which the Access Template is applied. Access Template link information is not inherited by any descendents of the object.
- 'ThisObjectAndAllChildObjects' - Indicates inheritance that includes the object to which the Access Template is applied, the object's immediate children, and the descendents of the object's children.
- 'ThisObjectAndImmediateChildObjects' - Indicates inheritance that includes the object itself and its immediate children. It does not include the descendents of its children.
- 'AllChildObjects' - Indicates inheritance that includes the object's immediate children and the descendants of the object's children, but not the object itself.
- 'ImmediateChildObjects' - Indicates inheritance that includes the object's immediate children only, not the object itself or the descendents of its children.

Confirm

Prompts you for confirmation before executing the command.

Description

Set the 'description' attribute to this parameter value.

Disabled

Supply this parameter for the cmdlet to configure the link to have no effect in ActiveRoles Server (disabled link).

Enabled

Supply this parameter for the cmdlet to configure the link to have effect in ActiveRoles Server (enabled link).

Identity

You can specify the name, Canonical Name, or Distinguished Name of the Access Template link (so as to identify the respective object located in the 'Configuration/AT Links' container in the ActiveRoles Server Configuration namespace).

Normally, pipelining is used to identify links: pass the output of the appropriate Get- cmdlet to this cmdlet. If you do so, the Identity parameter should not be supplied on the command line.

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet sets the specified attributes to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to set. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

SynchronizedToAD

Set this parameter to 'true' for the cmdlet to configure the link so as to propagate permission settings to Active Directory. To disable the propagation of the permission settings that result from the link, set this parameter to 'false'.

Trustee

Specify the identity (such as name, Distinguished Name, Domain\Name, SID, etc.) of a security principal (such as a user or group) you want. The cmdlet configures the link to determine access rights of that security principal (sets the specified object as the Trustee).

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to modify existing links of Access Templates in ActiveRoles Server. This cmdlet takes Access Template links returned by the respective Get- cmdlet, makes changes to the link data, and commits the changes to ActiveRoles Server. Each Access Template link contains information on how a certain Access Template is applied to determine access rights of a certain security principal (Trustee) on a certain directory object (securable object). For background information about Access Templates, see ActiveRoles Server Administrator Guide.

Examples

Example 1

Connect to any available Administration Service. Then, for every Access Template link on a given ActiveRoles Server Managed Unit, set a particular group as Trustee. This ensures that only members of that group have access to that Managed Unit in ActiveRoles Server:

```
C:\PS> connect-QADService -Proxy
C:\PS> get-QARSAccessTemplateLink -DirectoryObject 'Configuration/Managed
Units/ManagedUnitName' -Predefined $false | set-QARSAccessTemplateLink -Trustee
'DomainName\GroupName' | out-Null
```

Example 2

For a given organizational unit (OU) and a given Access Template applied on that OU, ensure that the permission settings defined by the Access Template on any object held in the OU are synchronized to Active Directory (on the respective Access Template link, enable the options to synchronize permission settings to AD and to apply them on both the OU and all child objects):

```
C:\PS> connect-QADService -Proxy
C:\PS> get-QADObject 'OrganizationalUnitName' -Type organizationalUnit |
%{get-QARSAccessTemplateLink -AccessTemplate 'AccessTemplateName' -DirectoryObject $_
-Predefined $false} | set-QARSAccessTemplateLink -SynchronizedToAD $true -AppliedTo
'ThisObjectAndAllChildObjects' | out-Null
```

Example 3

For a given organizational unit (OU) and a given Access Template, ensure that the permission settings defined by the Access Template on any object held in the OU are not synchronized to Active Directory (disable the permission synchronization option for each link that is based on that Access Template and applied to any object held in that OU):

```
C:\PS> connect-QADService -Proxy
C:\PS> get-QADObject -SearchRoot 'OrganizationalUnitName' | %{get-QARSAccessTemplateLink
-AccessTemplate 'AccessTemplateName' -DirectoryObject $_ -SynchronizedToAD $true
-Predefined $false} | set-QARSAccessTemplateLink -SynchronizedToAD $false | out-Null
```

New-QARSAccessTemplateLink

Use this cmdlet to apply ActiveRoles Server Access Templates.

Syntax

```
New-QARSAccessTemplateLink [[-Name] <String>] [-ObjectAttributes <Object>]
-AccessTemplate <IdentityParameter> -DirectoryObject <IdentityParameter> -Trustee
<IdentityParameter> [-Description <String>] [-AppliedTo <ATLinkFlags>]
[-SynchronizedToAD] [-Disabled] [-Proxy] [-UseGlobalCatalog] [-Service <String>]
[-ConnectionAccount <String>] [-ConnectionPassword <SecureString>] [-Credential
<PSCredential>] [-Connection <ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Note that this cmdlet requires a connection to the ActiveRoles Server Administration Service, so the *Proxy* parameter must be used to establish a connection.

Parameters

AccessTemplate

Specify the identity (such as name, Distinguished Name, etc.) of an Access Template you want. The cmdlet configures the link to apply that Access Template.

AppliedTo

Depending on the cmdlet you use, this parameter lets you either retrieve or configure Access Template links that have specific settings for permission inheritance. Valid parameter values are:

- 'This' - Indicates no inheritance. The Access Template link information is only used on the object to which the Access Template is applied. Access Template link information is not inherited by any descendents of the object.
- 'ThisObjectAndAllChildObjects' - Indicates inheritance that includes the object to which the Access Template is applied, the object's immediate children, and the descendents of the object's children.
- 'ThisObjectAndImmediateChildObjects' - Indicates inheritance that includes the object itself and its immediate children. It does not include the descendents of its children.
- 'AllChildObjects' - Indicates inheritance that includes the object's immediate children and the descendants of the object's children, but not the object itself.
- 'ImmediateChildObjects' - Indicates inheritance that includes the object's immediate children only, not the object itself or the descendents of its children.

Confirm

Prompts you for confirmation before executing the command.

Description

Set the 'description' attribute to this parameter value.

DirectoryObject

Specify the identity (such as name, Distinguished Name, Domain\Name, etc.) of a directory object you want. The cmdlet configures the link to apply the Access Template to that object (to determine security settings on that object).

Disabled

Supply this parameter for the cmdlet to configure the link to have no effect in ActiveRoles Server (disabled link).

Name

Optionally, specify a name for the link to create. If you omit this parameter, a name is auto-generated.

ObjectAttributes

Specify an associative array that defines the object attributes and values you want. The cmdlet sets the specified attributes to the specified values. Array syntax:

```
@{attr1='val1';attr2='val2';...}
```

In this syntax, each of the key-value pairs is the LDAP display name and the value of an attribute to set. For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

SynchronizedToAD

Set this parameter to 'true' for the cmdlet to configure the link so as to propagate permission settings to Active Directory. To disable the propagation of the permission settings that result from the link, set this parameter to 'false'.

Trustee

Specify the identity (such as name, Distinguished Name, Domain\Name, SID, etc.) of a security principal (such as a user or group) you want. The cmdlet configures the link to determine access rights of that security principal (sets the specified object as the Trustee).

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Using this cmdlet, you can apply Access Templates in ActiveRoles Server. The operation of applying an Access Template boils down to creation of an Access Template link. This cmdlet can take Access Template objects returned by the respective Get- cmdlet and create Access Template links, thus applying the Access Templates. Each Access Template link contains information on how a certain Access Template is applied to determine access rights of a certain security principal (Trustee) on a certain directory object (securable object). For background information about Access Templates, see ActiveRoles Server Administrator Guide.

Examples

Example 1

Give a certain group full control access to a certain Managed Unit in ActiveRoles Server. This command applies the appropriate pre-defined Access Template, creating an Access Template link on the Managed Unit, with the given group set as Trustee. The default permission inheritance setting (ThisObjectAndAllChildObjects) causes the Access Template link information to be used on any object in the managed domains:

```
C:\PS> connect-QADService -Proxy
C:\PS> new-QARSAccessTemplateLink -AccessTemplate 'Configuration/Access
Templates/Active Directory/All Objects - Full Control' -DirectoryObject
'Configuration/Managed Units/ManagedUnitName' -Trustee 'DomainName\GroupName'
```

Example 2

Connect to any available Administration Service. Then, configure security settings in ActiveRoles Server so as to give any authenticated user read access to any object in the Active Directory domains that are registered with ActiveRoles Server (managed domains). This command applies the appropriate pre-defined Access Template, creating an Access Template link on each of the domainDNS objects representing the managed domains, with Authenticated Users set as Trustee. The default permission inheritance setting (ThisObjectAndAllChildObjects) causes the Access Template link information to be used on any object in the managed domains:

```
C:\PS> connect-QADService -Proxy
C:\PS> get-QADObject -SearchRoot 'CN=Active Directory' -Type 'domainDNS' |
%{new-QARSAccessTemplateLink -AccessTemplate 'Configuration/Access Templates/Active
Directory/All Objects - Read All Properties' -DirectoryObject $_ -Trustee 'Authenticated
Users'}
```

Remove-QARSAccessTemplateLink

Delete Access Template links in ActiveRoles Server.

Syntax

```
Remove-QARSAccessTemplateLink [-Identity] <IdentityParameter> [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Note that this cmdlet requires a connection to the ActiveRoles Server Administration Service, so the *Proxy* parameter must be used to establish a connection.

Parameters

Confirm

Prompts you for confirmation before executing the command.

Identity

You can specify the name, Canonical Name, or Distinguished Name of the Access Template link (so as to identify the respective object located in the 'Configuration/AT Links' container in the ActiveRoles Server Configuration namespace).

Normally, pipelining is used to identify links: pass the output of the appropriate Get- cmdlet to this cmdlet. If you do so, the Identity parameter should not be supplied on the command line.

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to delete existing links of Access Templates in ActiveRoles Server. This cmdlet takes Access Template links returned by the respective Get- cmdlet, and requests ActiveRoles Server to delete those links. Each Access Template link contains information on how a certain Access Template is applied to determine access rights of a certain security principal (Trustee) on a certain directory object (securable object). For background information about Access Templates, see ActiveRoles Server Administrator Guide.

Examples

Example 1

Connect to any available Administration Service, create a new Access Template link, and then delete the link you created:

```
C:\PS> connect-QADService -Proxy
C:\PS> new-QARSAccessTemplateLink newATLink -AccessTemplate 'Configuration/Access
Templates/Active Directory/All Objects - Full Control' -DirectoryObject 'CN=Active
Directory' -Trustee 'Authenticated Users'
C:\PS> remove-QARSAccessTemplateLink newATLink -Confirm
```

Example 2

Given the name of an Access Template, ensure that the Access Template is no longer applied to any object (delete all links that are based on that Access Template):

```
C:\PS> connect-QADService -Proxy
C:\PS> get-QARSAccessTemplateLink -AccessTemplate 'AccessTemplateName' -Predefined
$false | remove-QARSAccessTemplateLink -Confirm
```

Example 3

Given the name of an Access Template and the name of an organizational unit (OU), remove all security settings on that OU that are determined by that Access Template (remove all links that are based on the given Access Template and applied to the given OU):

```
C:\PS> connect-QADService -Proxy
C:\PS> get-QARSAccessTemplateLink -AccessTemplate 'AccessTemplateName' -DirectoryObject
'OrganizationalUnitName' -Predefined $false | remove-QARSAccessTemplateLink -Confirm
```

Example 4

Given the name of an Access Template and the pre-Windows 2000 name of a group, revoke all access rights from that group that are defined by using that Access Template (remove all links that are based on the given Access Template and point to the given group):

```
C:\PS> connect-QADService -Proxy
C:\PS> get-QARSAccessTemplateLink -AccessTemplate 'AccessTemplateName' -Trustee
'domainName\groupName' -Predefined $false | remove-QARSAccessTemplateLink -Confirm
```

Get-QARSOperation

Retrieve operation records from ActiveRoles Server. Each operation record represents a certain change request, whether pending or completed, in ActiveRoles Server.

Syntax

```
Get-QARSOperation [[-TargetObject] <IdentityParameter[]>] [-OperationID <String[]>]
[-InitiatedAfter <DateTimeParameter>] [-InitiatedBefore <DateTimeParameter>]
[-InitiatedOn <DayParameter>] [-InitiatedRecently <RelativeDateTimeParameter>]
[-CompletedAfter <DateTimeParameter>] [-CompletedBefore <DateTimeParameter>]
[-CompletedOn <DayParameter>] [-CompletedRecently <RelativeDateTimeParameter>]
[-OperationType <OperationType[]>] [-OperationStatus <OperationStatus[]>] [-InitiatedBy
<IdentityParameter[]>] [-InitiatedByMe] [-TargetObjectType <String[]>]
[-ParentContainer <IdentityParameter[]>] [-AttributesChanges <Hashtable>]
[-ChangedAttributes <String[]>] [-SizeLimit <Int32>] [-Proxy] [-UseGlobalCatalog]
[-Service <String>] [-ConnectionAccount <String>] [-ConnectionPassword <SecureString>]
[-Credential <PSCredential>] [-Connection <ArsConnection>]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Note that this cmdlet requires a connection to the ActiveRoles Server Administration Service, so the *Proxy* parameter must be used to establish a connection.

Parameters

AttributeChanges

Retrieve records relating to requests for changing the specified attributes to the specified values. You can use this parameter to limit your search based on particular changes requested or made to particular attributes. Parameter value is an associative array that specifies the attributes and values you want. Array syntax:

```
@{attr1='val1';attr2='val2';…}
```

In this syntax, each of the key-value pairs is the LDAP display name of an attribute and the value set on the attribute in accord with the change request. Examples:

```
@{description="Text"}
```
Search for requests that change Description to the text specified.

```
@{description="Text*"}
```
Search for requests that change Description to any value that begins with the text specified.

```
@{description="$null"}
```
Search for requests that clear Description.

When multiple attributes are specified, the search returns the records that involve changes to all those attributes (the search conditions are combined using a logical AND operator).

For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

ChangedAttributes

Retrieve records relating to change requests targeted at the attribute specified. Parameter value can be an array of strings, with each string representing the LDAP display name of an attribute (such as 'description' or 'sAMAccountName'). You can use this parameter to limit your search based on any changes requested or made to particular attributes. When multiple attributes are specified, the search returns the records that involve changes to any of those attributes (the search conditions are combined using a logical OR operator).

CompletedAfter

Retrieve operation records for the change requests that were completed after the date specified. Parameter value can be a DateTime object or a string that specifies the date you want. Thus, you can supply "Today" as a parameter value.

CompletedBefore

Retrieve operation records for the change requests that were completed before the date specified. Parameter value can be a DateTime object or a string that specifies the date you want. Thus, you can supply "Today" as a parameter value.

CompletedOn

Retrieve operation records for the change requests that were completed within the date specified. Parameter value can be a DateTime object or a string that specifies the date you want. Thus, you can supply "Today" as a parameter value.

CompletedRecently

Retrieve operation records for the change requests that were completed during the recent time period specified. Parameter value is a TimeSpan object that specifies the time period you want. For example, if you supply a time span of 3 days, the cmdlet searches for the change requests completed during the last 3 days.

InitiatedAfter

Retrieve operation records for the change requests that occurred after the date specified. Parameter value can be a DateTime object or a string that specifies the date you want. Thus, you can supply "Today" as a parameter value.

InitiatedBefore

Retrieve operation records for the change requests that occurred before the date specified. Parameter value can be a DateTime object or a string that specifies the date you want. Thus, you can supply "Today" as a parameter value.

InitiatedBy

Retrieve operation records for the changes that were requested by the security principal (such as a user) specified. Parameter value can be an array of strings, with each string representing the Distinguished Name (DN), Canonical Name, Domain\Name, User Principal Name, SID or GUID of a security principal object in Active Directory. You can use this parameter to retrieve information about changes to directory data made by a particular person (user activity).

InitiatedByMe

Retrieve operation records for the changes that were requested by the security principal (such as a user) in whose security context the cmdlet is currently running.

InitiatedOn

Retrieve operation records for the change requests that occurred within the date specified. Parameter value can be a DateTime object or a string that specifies the date you want. Thus, you can supply "Today" as a parameter value.

InitiatedRecently

Retrieve operation records for the change requests that occurred during the recent time period specified. Parameter value is a TimeSpan object that specifies the time period you want. For example, if you supply a time span of 3 days, the cmdlet searches for the records regarding change requests that occurred during the last 3 days.

OperationID

Retrieve an operation record by ID. Parameter value is the string ID of the operation to retrieve (you can view operation ID in the Approval section of the ActiveRoles Server Web Interface).

OperationStatus

Retrieve operation records for the change requests that are completed (the changes are applied or rejected) or pending (awaiting approval). Parameter value can be one of the following:

- Canceled
- Completed
- Denied
- Failed
- InProgress
- Pending

If this parameter is omitted, operation records are retrieved for any requests.

OperationType

Retrieve records for the change requests of the category specified. Valid parameter values are:

- Create
- Delete
- Copy
- Modify
- GroupMembershipChange

- Move
- Rename
- Deprovision
- UndoDeprovision

Parameter value can be any combination of the listed values, separated by commas. For example, 'Create,Modify' limits the search to the requests for changing attributes of existing objects or creation of new objects. If this parameter is omitted, records are retrieved regardless of the change request category.

ParentContainer

Retrieve operation records for the change requests targeted at directory objects that reside in the container (such as an organizational unit) specified. Parameter value can be an array of strings, with each string representing the Distinguished Name (DN), Canonical Name, or GUID of a container object in the directory. You can use this parameter to retrieve information about changes made to directory object in a particular container.

SizeLimit

Set the maximum number of items to be returned by the cmdlet. Normally, the default size limit is 1000. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

TargetObject

Retrieve operation records for the change requests targeted at the directory object specified. Parameter value can be an array of strings, with each string representing the Distinguished Name (DN), Canonical Name, Domain\Name, User Principal Name, SID or GUID of a directory object. You can use this parameter to retrieve information about changes made to a particular object in the directory (change history).

TargetObjectType

Retrieve operation records for the change requests targeted at the directory objects of the object class specified. Parameter value can be an array of strings, with each string representing the LDAP display name of an object class (such as 'user' or 'organizationalUnit'). You can use this parameter to retrieve information about changes made to directory objects of a particular type.

Detailed Description

Use this cmdlet to retrieve information about requests to make changes to directory data using ActiveRoles Server. Each request causes an operation record to be created in the ActiveRoles Server Management History data store. The cmdlet provides access to operation records, allowing you to examine:

- **Change History** - Information on changes that were made to particular pieces of directory data (directory objects and their attributes) using ActiveRoles Server.
- **User Activity** - Information on directory data changes that were made by particular users.

For a discussion of the Management History feature, see "Management History" in the ActiveRoles Server Administrator Guide.

An operation record may represent a change request that is waiting for approval in ActiveRoles Server. Such operation records are created whenever the requested changes require approval in accord with the ActiveRoles Server approval rules, and include information about the associated approval tasks. Once all approval tasks associated with an operation are completed, the respective operation record is marked completed and the requested changes are either applied or denied depending the approvers' resolutions on the approval tasks (Approve or Reject). For more information about the change approval function, see "Approval Workflow" in the ActiveRoles Server Administrator Guide.

The objects returned by this cmdlet represent operation records that meet the search conditions specified, allowing you to access information about the respective change requests. You can discover what changes were requested, when and by whom change requests were initiated, when the requested changes were applied, as well as get identities of operation records for further processing such as retrieval and accomplishment of approval tasks.

Return Type

Type of object returned by the cmdlet:
Quest.ActiveRoles.ArsPowerShellSnapIn.Output.Operation

Quest.ActiveRoles.ArsPowerShellSnapIn.Output.Operation Properties

Objects of the Output.Operation object type contain information about ActiveRoles Server operation records, representing change requests in the ActiveRoles Server Management History data store.

This object type exposes the following properties:

PROPERTY	DESCRIPTION
ID Syntax: <string>	An identifier of the operation record.
OperationGuid Syntax: <Guid>	Globally unique identifier of the operation record.
Type Syntax: <Enumeration>	Identifies the category f the change request represented by the operation record. Can be one of the following: • Create • Delete • Copy • Modify • GroupMembershipChange • Move • Rename • Deprovision • UndoDeprovision

Status Syntax: <Enumeration>	Indicates whether the status of the change request, such as whether the changes are applied or pending (awaiting approval). Property value can be one of the following: • Canceled • Completed • Denied • Failed • InProgress • Pending
InitiatorInfo Syntax: <Output.PrincipalInformation>	Object containing information about the identity (for example, a user) that requested the changes. The properties exposed by this object type are listed later in this section.
TargetObjectInfo Syntax: <Output.ObjectInformation>	Object containing information about the directory object (for example, a user account or a group) to which the changes were requested. The properties exposed by this object type are listed later in this section.
Initiated Syntax: <DateTime>	Date and time that the changes were requested.
Completed Syntax: <DateTime>	Date and time that the change request was completed (the changes were applied or rejected).
TasksCount Syntax: <int>	Number of the approval tasks associated with the change request.
AttributeChanges Syntax: <AttributeChangeInfo[]>	Collection of objects containing information on the attribute changes that were requested. The properties exposed by this object type are listed later in this section.
Controls Syntax: <ControlInfo[]>	Collection of objects containing information about the control codes that were added to the change request. This object type exposes two properties: Name and Value, identifying the name and the value of the control code, respectively. For information about ActiveRoles Server control codes, see the ActiveRoles Server SDK and Resource Kit documentation.
NewObjectDN Syntax: <string>	For a change request of the Rename or Move category, identifies the Distinguished Name of the operation target object after applying the requested changes.
SourceObjectDN Syntax: <string>	For a change request of the Copy category (creation of a new object by copying an existing object), identifies the Distinguished Name of the object to copy.

Quest.ActiveRoles.ArsPowerShellSnapIn.Output.PrincipalInformation Properties

Objects of the Output.PrincipalInformation object type contain information about Active Directory security principal objects (such as users or groups).

This object type exposes the following properties:

PROPERTY	DESCRIPTION
DN Syntax: <string>	Distinguished Name of the security principal object in Active Directory.

Guid Syntax: <Guid>	GUID of the security principal object in Active Directory.
Sid Syntax: <Principal.SecurityIdentifier>	SID of the security principal object in Active Directory.
NTAccountName Syntax: <string>	Pre-Windows 2000 name of the security principal object, in the form of Domain\Name where Domain is the NetBIOS name of the Active Directory domain in which the object is defined.
ObjectClass Syntax: <string>	Identifies the type of the security principal object (such as User or Group).
Host Syntax: <string>	Identifies the computer running the ActiveRoles Server client application used by the security principal.
Site Syntax: <string>	Identifies the site of the computer running the ActiveRoles Server client application used by the security principal.
IsDSAdmin Syntax: <bool>	Indicates whether the security principal is an AR Server Admin (DS Admin) role holder, and thus has full access to ActiveRoles Server.

Quest.ActiveRoles.ArsPowerShellSnapIn.Output.ObjectInformation Properties

Objects of the Output.ObjectInformation object type contain information about directory objects.

This object type exposes the following properties:

PROPERTY	DESCRIPTION
DN Syntax: <string>	Distinguished Name of the directory object.
Guid Syntax: <Guid>	GUID of the directory object.
Sid Syntax: <Principal.SecurityIdentifier>	SID of the directory object (only applies to security principal objects).
NTAccountName Syntax: <string>	Pre-Windows 2000 name of the directory object (only applies to security principal objects), in the form of Domain\Name where Domain is the NetBIOS name of the Active Directory domain in which the object is defined.
ObjectClass Syntax: <string>	Identifies the type of the directory object (such as User or Group).

AttributeChangeInfo Properties

An object of the AttributeChangeInfo object type contains information about changes to a single attribute in the directory.

This object type exposes the following properties:

PROPERTY	DESCRIPTION
Name Syntax: <string>	Identifies the attribute by name.
Type Syntax: <Enumeration>	Data type (syntax) of the attribute. Can be one of the following: • Unspecified • String • Boolean • Integer • LargeInteger • DateTime • OctetString
Operation Syntax: <Enumeration>	Category of changes to the attribute. Can be one of the following: • Add • Delete • Replace
SerializedValues Syntax: <string[]>	Array of strings representing the new attribute values in a serialized form. The attribute is changed by applying these values in accord with the category of changes (see Operation property).

Examples

Example 1

List the user accounts from a particular container that were changed on the current date:

```
C:\PS> Get-QARSOperation -CompletedOn 'Today' -ParentContainer
'test.domain.com/container' -TargetObjectType 'user' -OperationType 'Modify' |
%{$_.TargetObjectInfo.DN} | Group-Object | %{$_.Name}
```

Example 2

List the groups that were created in a particular container on September 1, 2008:

```
C:\PS> Get-QARSOperation -CompletedOn (get-date -year 2008 -month 9 -day 1)
-ParentContainer 'test.domain.com/container' -TargetObjectType 'Group' -OperationType
'Create' | %{$_.TargetObjectInfo.DN} | Group-Object | %{$_.Name}
```

Example 3

List the names of the security principals that added or removed members from a particular group during last month:

```
C:\PS> Get-QARSOperation -CompletedRecently ([TimeSpan]::FromDays(30)) -TargetObject
'domainName\groupName' -OperationType 'GroupMembershipChange' |
%{$_.InitiatorInfo.NTAccountName} | Group-Object | %{$_.Name}
```

Example 4

List the names of the security principals that changed or reset the password of a particular user account:

```
C:\PS> Get-QARSOperation -TargetObject 'domain\user' -OperationType 'Modify'
-ChangedAttributes 'edsaPassword' | %{$_.InitiatorInfo.NTAccountName} | Group-Object |
select Name
```

Example 5

List all user accounts from a particular container that were changed by the user 'MyDomain\JSmith' during last week:

```
C:\PS> Get-QARSOperation -CompletedRecently ([TimeSpan]::FromDays(7)) -TargetObjectType
user -ParentContainer 'test.domain.com/container' -InitiatedBy 'MyDomain\JSmith' |
%{$_.TargetObjectInfo.DN}
```

Example 6

List the names of the security principals that changed the City (l) or Street Address (streetAddress) attribute on the account of a particular user account yesterday:

```
C:\PS> Get-QARSOperation -TargetObject 'domain\user' -ChangedAttributes l,streetAddress
-CompletedOn ((get-date).AddDays(-1)) | %{$_.InitiatorInfo.NTAccountName} | Group-Object
| select Name
```

Example 7

List the groups from a particular container that had the membership list (Members attribute) changed during the time period from September 15, 2008 to September 30, 2008:

```
C:\PS> Get-QARSOperation -ParentContainer test.domain.com/container -TargetObjectType
group -OperationType 'GroupMembershipChange' -CompletedAfter (get-date -year 2008
-month 9 -day 15 -hour 0 -minute 0 -second 0) -CompletedBefore (get-date -year 2008
-month 9 -day 30 -hour 23 -minute 59 -second 59) | %{$_.TargetObjectInfo.DN} |
Group-Object | %{$_.Name}
```

Get-QARSApprovalTask

Retrieve approval task records from ActiveRoles Server. Each approval task record represents a task, whether pending or completed, to approve or reject a certain change request in ActiveRoles Server.

Syntax

```
Get-QARSApprovalTask [-TaskID <String[]>] [-Operation <OperationIdentityParameter[]>]
[-CreatedAfter <DateTimeParameter>] [-CreatedBefore <DateTimeParameter>] [-CreatedOn
<DayParameter>] [-CreatedRecently <RelativeDateTimeParameter>] [-CompletedAfter
<DateTimeParameter>] [-CompletedBefore <DateTimeParameter>] [-CompletedOn
<DayParameter>] [-CompletedRecently <RelativeDateTimeParameter>] [-OperationType
<OperationType[]>] [-TaskStatus <TaskStatus[]>] [-Approver <IdentityParameter[]>]
[-ApproverIsMe] [-CompletedBy <IdentityParameter[]>] [-CompletedByMe]
[-AttributesChanges <Hashtable>] [-ChangedAttributes <String[]>] [-SizeLimit <Int32>]
[-Proxy] [-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Note that this cmdlet requires a connection to the ActiveRoles Server Administration Service, so the *Proxy* parameter must be used to establish a connection.

Parameters

Approver

Retrieve records for the approval tasks that have the specified security principal (such as a user or group) designated as an approver (a person authorized to approve or reject the respective change request). Parameter value can be an array of strings, with each string representing the Distinguished Name (DN), Canonical Name, Domain\Name, User Principal Name, SID or GUID of a security principal object in Active Directory. You can use this parameter to retrieve information about approval tasks assigned to particular persons.

ApproverIsMe

Retrieve records for the approval tasks that are assigned to the security principal (such as a user) in whose security context the cmdlet is currently running.

AttributeChanges

Retrieve records relating to requests for changing the specified attributes to the specified values. You can use this parameter to limit your search based on particular changes requested or made to particular attributes. Parameter value is an associative array that specifies the attributes and values you want. Array syntax:

```
@{attr1='val1';attr2='val2';…}
```

In this syntax, each of the key-value pairs is the LDAP display name of an attribute and the value set on the attribute in accord with the change request. Examples:

```
@{description="Text"}
```
Search for requests that change Description to the text specified.

```
@{description="Text*"}
```
Search for requests that change Description to any value that begins with the text specified.

```
@{description="$null"}
```
Search for requests that clear Description.

When multiple attributes are specified, the search returns the records that involve changes to all those attributes (the search conditions are combined using a logical AND operator).

For information about associative arrays, type the following command at the PowerShell command-prompt:

```
help about_associative_array
```

ChangedAttributes

Retrieve records relating to change requests targeted at the attribute specified. Parameter value can be an array of strings, with each string representing the LDAP display name of an attribute (such as 'description' or 'sAMAccountName'). You can use this parameter to limit your search based on any changes requested or made to particular attributes. When multiple attributes are specified, the search returns the records that involve changes to any of those attributes (the search conditions are combined using a logical OR operator).

CompletedAfter

Retrieve records for the approval tasks that were completed (that is, received the Approve or Reject resolution) after the date specified. Parameter value can be a DateTime object or a string that specifies the date you want. Thus, you can supply "Today" as a parameter value.

CompletedBefore

Retrieve records for the approval tasks that were completed (that is, received the Approve or Reject resolution) before the date specified. Parameter value can be a DateTime object or a string that specifies the date you want. Thus, you can supply "Today" as a parameter value.

CompletedBy

Retrieve records for the approval tasks that were completed by the security principal (such as a user) specified. Parameter value can be an array of strings, with each string representing the Distinguished Name (DN), Canonical Name, Domain\Name, User Principal Name, SID or GUID of a security principal object in Active Directory. You can use this parameter to retrieve information about approval tasks to which a particular person has applied the Approve or Reject resolution.

CompletedByMe

Retrieve records for the approval tasks that were completed by the security principal (such as a user) in whose security context the cmdlet is currently running.

CompletedOn

Retrieve records for the approval tasks that were completed (that is, received the Approve or Reject resolution) within the date specified. Parameter value can be a DateTime object or a string that specifies the date you want. Thus, you can supply "Today" as a parameter value.

CompletedRecently

Retrieve records for the approval tasks that were completed (that is, received the Approve or Reject resolution) during the recent time period specified. Parameter value is a TimeSpan object that specifies the time period you want. For example, if you supply a time span of 3 days, the cmdlet searches for the records regarding the approval tasks completed during the last 3 days.

CreatedAfter

Retrieve approval task records for the change requests that occurred after the date specified. Parameter value can be a DateTime object or a string that specifies the date you want. Thus, you can supply "Today" as a parameter value.

CreatedBefore

Retrieve approval task records for the change requests that occurred before the date specified. Parameter value can be a DateTime object or a string that specifies the date you want. Thus, you can supply "Today" as a parameter value.

CreatedOn

Retrieve approval task records for the change requests that occurred within the date specified. Parameter value can be a DateTime object or a string that specifies the date you want. Thus, you can supply "Today" as a parameter value.

CreatedRecently

Retrieve approval task records for the change requests that occurred during the recent time period specified. Parameter value is a TimeSpan object that specifies the time period you want. For example, if you supply a time span of 3 days, the cmdlet searches for the task records specific to the change requests that occurred during the last 3 days.

Operation

Retrieve approval task records that are associated with the operation records specified. Parameter value is an object or a collection of objects returned by the Get-QARSOperation cmdlet, or an array of operation string IDs. You can use this parameter to access information about approval tasks specific to a particular change request that requires approval in ActiveRoles Server.

OperationType

Retrieve records for the change requests of the category specified. Valid parameter values are:

- Create
- Delete

- Copy
- Modify
- GroupMembershipChange
- Move
- Rename
- Deprovision
- UndoDeprovision

Parameter value can be any combination of the listed values, separated by commas. For example, 'Create,Modify' limits the search to the requests for changing attributes of existing objects or creation of new objects. If this parameter is omitted, records are retrieved regardless of the change request category.

SizeLimit

Set the maximum number of items to be returned by the cmdlet. Normally, the default size limit is 1000. You can view or modify this default setting by using the Get- or Set-QADPSSnapinSettings cmdlet, respectively.

TaskID

Retrieve approval task records by ID. Parameter value is an array of strings, with each string representing the ID of the task to retrieve (you can view task ID in the Approval section of the ActiveRoles Server Web Interface).

TaskStatus

Retrieve records for the approval tasks that are completed with a certain resolution (Approve or Reject) or pending (awaiting resolution). Valid parameter values are:

- Pending
- Canceled
- CompletedApproved
- CompletedRejected

Parameter value can be any combination of the listed values, separated by commas. For example, 'CompletedApproved,CompletedRejected' limits the search to the tasks that have been completed with any resolution. If this parameter is omitted, records are retrieved regardless of the tasks status (a search is performed against all tasks).

Detailed Description

Use this cmdlet to retrieve information about tasks of approving changes to directory data that were requested using ActiveRoles Server.

When changes are attempted that require approval, one or more approval task records are created in the ActiveRoles Server Management History data store, and associated with the operation record representing the change request.

Once all approval tasks associated with an operation are completed, the operation record is marked completed and the requested changes are either applied or denied depending the approvers' resolutions on the approval tasks (Approve or Reject). When any task associated with an operation receives the Reject resolution, the entire operation is denied and the requested changes are disregarded. When all tasks associated with the operation receive the Approve resolution, the operation is allowed and the requested changes are applied. For more information on the change approval function, see "Approval Workflow" in the ActiveRoles Server Administrator Guide.

The objects returned by this cmdlet represent approval task records that meet the search condition specified, allowing you to access information about the approval tasks and the operation they govern. You can discover who is expected to perform approval tasks, when and by whom the requested changes were approved or rejected, as well as get identities of approval task records for further processing in order to perform approval tasks.

Return Type

Type of object returned by the cmdlet:
Quest.ActiveRoles.ArsPowerShellSnapIn.Output.ApprovalTask

Quest.ActiveRoles.ArsPowerShellSnapIn.Output.ApprovalTask Properties

Objects of the Output.ApprovalTask object type contain information about ActiveRoles Server approval task records, representing tasks to approve or reject change requests in ActiveRoles Server.

This object type exposes the following properties:

PROPERTY	DESCRIPTION
ID Syntax: <string>	An identifier of the approval task record.
TaskGuid Syntax: <Guid>	Globally unique identifier of the approval task record.
Status Syntax: <Enumeration>	Indicates the status of the approval task, such as whether the task is completed (the approver has applied the Approve or Reject resolution on the task) or pending (awaiting an approver's action). Property value can be one of the following. • Pending • Canceled • Completed • CompletedApproved • CompletedRejected • Reviewed
ApprovalRuleGuid Syntax: <Guid>	Identifies the ActiveRoles Server approval rule based on which the approval task was created.
Created Syntax: <DateTime>	Date and time that the approval task record was created.
LastChanged Syntax: <DateTime>	Date and time that the approval task was last changed.

Completed Syntax: <DateTime>	Date and time that the approval task was completed.
CompletedBy Syntax: <Output.PrincipalInformation>	Object containing information about the identity (such as a user) who completed the task (applied the Approve or Reject resolution on the task). The properties exposed by this object type are listed in the section that discusses the Get-QARSOperation cmdlet.
CompletionReason Syntax: <string>	Comment text that was entered by the approver upon completion of the task.
ApproversInfo Syntax: <Output.ObjectInformation[]>	Collection of objects containing information about the approvers assigned to the task (the identities that are expected to perform the approval task). The properties exposed by this object type are listed in the section that discusses the Get-QARSOperation cmdlet.
Operation Syntax: <Output.Operation>	Object containing information about the operation (change request) with which the approval task is associated. The properties exposed by this object type are listed in the section that discusses the Get-QARSOperation cmdlet.

Examples

Example 1

List all approval tasks awaiting response from the user 'MyDomain\JSmith':

```
C:\PS> Get-QARSApprovalTask -Approver 'MyDomain\JSmith'
```

Example 2

List all change requests (operations) awaiting approval by the current user:

```
C:\PS> Get-QARSApprovalTask -ApproverIsMe -TaskStatus Pending | %{$_.Operation}
```

Example 3

List all change requests (operations) awaiting approval by the user 'MyDomain\JSmith':

```
C:\PS> Get-QARSApprovalTask -Approver 'MyDomain\JSmith' -TaskStatus Pending |
%{$_.Operation}
```

Example 4

List all change requests (operations) that were approved by the user 'MyDomain\JSmith' during last week:

```
C:\PS> Get-QARSApprovalTask -CompletedRecently ([TimeSpan]::FromDays(7)) -CompletedBy
'MyDomain\JSmith' -TaskStatus CompletedApproved | %{$_.Operation}
```

Example 5

List all approval tasks that the user 'MyDomain\JSmith' completed with the Reject resolution during last week:

```
C:\PS> Get-QARSApprovalTask -CompletedRecently ([TimeSpan]::FromDays(7)) -CompletedBy
'MyDomain\JSmith' -TaskStatus CompletedRejected
```

Approve-QARSApprovalTask

Apply the Approve resolution on approval tasks for which you are assigned to the Approver role in ActiveRoles Server.

Syntax

```
Approve-QARSApprovalTask [-Task] <TaskIdentityParameter> [-Reason <String>] [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Note that this cmdlet requires a connection to the ActiveRoles Server Administration Service, so the *Proxy* parameter must be used to establish a connection.

Parameters

Confirm

Prompts you for confirmation before executing the command.

Reason

Supply any text you want to comment your decision. This text is saved by ActiveRoles Server for reporting and audit purposes, and can be accesses through the respective property of the ApprovalTask object.

Task

Pass an object returned by the Get-QARSApprovalTask cmdlet to this parameter. This parameter also accepts the numeric ID of an approval task (you can view task ID in the Approval section of the ActiveRoles Server Web Interface).

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to approve change requests that require your approval in ActiveRoles Server. You can take an object returned by the Get-QARSApprovalTask cmdlet and pipe that object into this cmdlet in order to complete the respective approval task with the Approve resolution.

Every change request that requires your approval has an approval task for you to allow or deny the requested changes. When you complete your approval task with the Approve resolution, you allow the changes. Note that the changes may also require approval by other persons. In this case, the changes are applied only after all approval tasks (including yours) are completed with the Approve resolution. For more information about approval tasks, see description of the Get-QARSApprovalTask cmdlet.

Return Type

Type of object returned by the cmdlet:
Quest.ActiveRoles.ArsPowerShellSnapIn.Output.ApprovalTask

The properties exposed by this object type are listed in the section that discusses the Get-QARSApprovalTask cmdlet.

Examples

Example 1

Approve all changes that are awaiting approval by the current user:

```
C:\PS> Get-QARSApprovalTask -ApproverIsMe -TaskStatus Pending | Approve-QARSApprovalTask
```

Example 2

Approve all changes to a particular user account that are awaiting approval by the current user:

```
C:\PS> Get-QARSOperation -TargetObject 'domainName\userName' -OperationStatus Pending |
%{Get-QARSApprovalTask -Operation $_ -ApproverIsMe} | Approve-QARSApprovalTask
```

Reject-QARSApprovalTask

Apply the Reject resolution on approval tasks for which you are assigned to the Approver role in ActiveRoles Server.

Syntax

```
Reject-QARSApprovalTask [-Task] <TaskIdentityParameter> -Reason <String> [-Proxy]
[-UseGlobalCatalog] [-Service <String>] [-ConnectionAccount <String>]
[-ConnectionPassword <SecureString>] [-Credential <PSCredential>] [-Connection
<ArsConnection>] [-WhatIf] [-Confirm]
```

The cmdlet has optional parameters that determine the server and the security context for the operation. The connection parameters could be omitted since a connection to a server is normally established prior to using this cmdlet. In this case, the server and the security context are determined by the **Connect-QADService** cmdlet. If you do not use **Connect-QADService** and have no connection established prior to using a cmdlet, then the connection settings, including the server and the security context, are determined by the connection parameters of the first cmdlet you use. Subsequent cmdlets will use those settings by default.

The connection parameters include: *Proxy*, *Service*, *ConnectionAccount*, *ConnectionPassword*, *Credential*, *Connection*, and *UseGlobalCatalog*. For parameter descriptions, see the "Connect-QADService" section earlier in this document.

Note that this cmdlet requires a connection to the ActiveRoles Server Administration Service, so the *Proxy* parameter must be used to establish a connection.

Parameters

Confirm

Prompts you for confirmation before executing the command.

Reason

Supply any text you want to comment your decision. This text is saved by ActiveRoles Server for reporting and audit purposes, and can be accesses through the respective property of the ApprovalTask object.

Task

Pass an object returned by the Get-QARSApprovalTask cmdlet to this parameter. This parameter also accepts the numeric ID of an approval task (you can view task ID in the Approval section of the ActiveRoles Server Web Interface).

WhatIf

Describes what would happen if you executed the command, without actually executing the command.

Detailed Description

Use this cmdlet to reject change requests that require your approval in ActiveRoles Server. You can take an object returned by the Get-QARSApprovalTask cmdlet and pipe that object into this cmdlet in order to complete the respective approval task with the Reject resolution.

Every change request that requires your approval has an approval task for you to allow or deny the requested changes. When you complete your approval task with the Reject resolution, you effectively deny the changes, preventing them from being applied. Note that the changes may also require approval by other persons. In this case, the changes are not applied if any one of the approval tasks is completed with the Reject resolution. For more information about approval tasks, see description of the Get-QARSApprovalTask cmdlet.

Return Type

Type of object returned by the cmdlet:
Quest.ActiveRoles.ArsPowerShellSnapIn.Output.ApprovalTask

The properties exposed by this object type are listed in the section that discusses the Get-QARSApprovalTask cmdlet.

Examples

Example 1

Reject all changes to a particular user account that were requested on the current date and are awaiting approval by the current user:

```
C:\PS> Get-QARSOperation -TargetObject 'domainName\userName' -InitiatedOn 'Today'
-OperationStatus Pending | %{Get-QARSApprovalTask -Operation $_ -ApproverIsMe} |
Reject-QARSApprovalTask
```

Example 2

Reject all changes to a particular user account that were requested by the user 'MyDomain\JohnSmith' and are awaiting approval by the current user:

```
C:\PS> Get-QARSOperation -TargetObject 'domainName\userName' -InitiatedBy
'MyDomain\JohnSmith' -OperationStatus Pending | %{Get-QARSApprovalTask
-Operation $_ -ApproverIsMe} | Reject-QARSApprovalTask
```

Cmdlet Reference - Utility

Here you can find information about command-line tools (cmdlets) that are provided by ActiveRoles Management Shell.

This section covers the utility cmdlets, such as cmdlets for configuring the shell or converting data from one data type to another.

Convert-QADAttributeValue

Convert attribute values of a directory object to the specified .NET type.

Syntax

```
Convert-QADAttributeValue -Input <Object> -OutputTypeName <String>
```

Parameters

Input

Specify the object representing the attribute value to convert. This parameter accepts pipeline input, and can be omitted on the command line if you pipe into this cmdlet an object returned by a Get- cmdlet (see examples).

OutputTypeName

Specify the fully qualified name of the .NET type to convert the attribute value to. The assembly name and namespace indication can be omitted if the type is from the System namespace (see examples).

Detailed Description

Use this cmdlet to convert attribute values of directory objects returned by a cmdlet (for example, by a **Get-QADUser** cmdlet). You can convert:

- Values of the byte[] type to the SecurityIdentifier or Guid type
- Values of the IADsLargeInteger type to the Int64, DateTime, or TimeSpan type

Examples

Example 1

Convert the value of the objectGuid attribute to the Guid type, and display the value in the console window:

```
C:\PS> get-QADuser 'MyDomain\JSmith' | %{$_.DirectoryEntry.objectGuid} |
convert-QADAttributeValue -outputTypeName 'Guid' | Write-Host
```

Example 2

Convert the value of the objectSid attribute to the SecurityIdentifier type, and display the value in the console window:

```
C:\PS> get-QADuser 'MyDomain\JSmith' | %{$_.DirectoryEntry.objectSid} |
convert-QADAttributeValue -outputTypeName 'Security.Principal.SecurityIdentifier' |
Write-Host
```

Example 3

Convert the value of the lastLogon attribute to the DateTime type, and display the value in the console window:

```
C:\PS> get-QADuser 'MyDomain\JSmith' | %{$_.DirectoryEntry.lastLogon} |
convert-QADAttributeValue -outputTypeName 'DateTime' | Write-Host
```

Example 4

For each domain controller, retrieve the time that the user JSmith last logged on by using a particular domain controller, and display the results in the console window:

```
C:\PS> get-QADComputer -searchRoot 'mydomain.company.com/domain controllers' |
Select-Object Name,@{Name="Last Logon"; Expression={%{ get-QADUser 'MyDomain\JSmith'
-Service $_.Name} | %{$_.DirectoryEntry.lastLogon} | convert-QADAttributeValue
-outputTypeName 'DateTime'}}
```

Get-QADPSSnapinSettings

View default settings that apply to all cmdlets of this PowerShell snap-in.

Syntax

```
Get-QADPSSnapinSettings [-DefaultExcludedProperties]
[-DefaultPropertiesExcludedFromNonBaseSearch] [-Integer8AttributesThatContainDateTimes]
[-Integer8AttributesThatContainNegativeTimeSpans] [-DefaultPageSize]
[-DefaultSizeLimit] [-DefaultSearchScope] [-DefaultWildcardMode]
[-DefaultOutputPropertiesForUserObject] [-DefaultOutputPropertiesForGroupObject]
[-DefaultOutputPropertiesForComputerObject] [-DefaultOutputPropertiesForAdObject]
[-DefaultOutputPropertiesForPasswordSettingsObject]
```

Parameters

DefaultExcludedProperties

This parameter causes the cmdlet to return a list of the attributes that are excluded from processing by the UseDefaultExcludedProperties parameter on any particular cmdlet.

DefaultOutputPropertiesForAdObject

This parameter causes the cmdlet to return the default list of the object attributes that are retrieved from the directory and stored in the local memory cache by a Get- cmdlet during a search for objects other than a User, Group, Computer, or Password Settings object.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by a Get- cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

DefaultOutputPropertiesForComputerObject

This parameter causes the cmdlet to return the default list of the Computer object attributes that are retrieved from the directory and stored in the local memory cache by a Get- cmdlet during a search for Computer objects.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by a Get- cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

DefaultOutputPropertiesForGroupObject

This parameter causes the cmdlet to return the default list of the Group object attributes that are retrieved from the directory and stored in the local memory cache by a Get- cmdlet during a search for Group objects.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by a Get- cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

DefaultOutputPropertiesForPasswordSettingsObject

This parameter causes the cmdlet to return the default list of the Password Settings object attributes that are retrieved from the directory and stored in the local memory cache by a Get- cmdlet during a search for Password Settings objects.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by a Get- cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

DefaultOutputPropertiesForUserObject

This parameter causes the cmdlet to return the default list of the User object attributes that are retrieved from the directory and stored in the local memory cache by a Get- cmdlet during a search for User objects.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by a Get- cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

DefaultPageSize

Supply this parameter to view the default value of the PageSize parameter that is used by the Get- cmdlets. This page size value is used if the PageSize parameter is omitted.

DefaultPropertiesExcludedFromNonBaseSearch

This parameter causes the cmdlet to return a list of the attributes that are not retrieved from the directory and stored in the local memory cache by any particular Get- cmdlet during a search with the search scope other than 'Base'.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

DefaultSearchScope

Supply this parameter to view the default value of the SearchScope parameter that is used by the Get- cmdlets. This search scope setting is used if the SearchScope parameter is omitted.

DefaultSizeLimit

Supply this parameter to view the default value of the SizeLimit parameter that is used by the Get- cmdlets. This limitation on the size of the search result set is used if the SizeLimit parameter is omitted.

DefaultWildcardMode

Supply this parameter to view the default value of the WildcardMode parameter that is used by the Get- cmdlets. This wildcard mode setting is used if the WildcardMode parameter is omitted.

Integer8AttributesThatContainDateTimes

This parameter causes the cmdlet to return a list of the Integer8 attributes whose values are represented as DateTime objects in the output of the Get- cmdlets by default. Each attribute is identified by its LDAP display name.

Note: This setting applies only to the properties of a cmdlet's output object that have the member type of NoteProperty. Such properties are normally added to the output object in order to provide access to the attribute values of the respective directory object that are loaded to the local memory cache but cannot be accessed by using properties of the base object (the object for which the output object serves as a wrapper).

Integer8AttributesThatContainNegativeTimeSpans

This parameter causes the cmdlet to return a list of the Integer8 attributes whose values are represented as TimeSpan objects in the output of the Get- cmdlets by default. Each attribute is identified by its LDAP display name.

The output TimeSpan objects represent the absolute values of the attributes, and thus indicate positive time intervals regardless of whether an actual attribute value is a negative or positive time interval.

Note: This setting applies only to the properties of a cmdlet's output object that have the member type of NoteProperty. Such properties are normally added to the output object in order to provide access to the attribute values of the respective directory object that are loaded to the local memory cache but cannot be accessed by using properties of the base object (the object for which the output object serves as a wrapper).

Detailed Description

You can use this cmdlet to view some default settings that have effect within this PowerShell snap-in on any cmdlet where those settings are applicable. To change default settings, use the **Set-QADPSSnapinSettings** cmdlet.

Set-QADPSSnapinSettings

Modify default settings that apply to all cmdlets of this PowerShell snap-in.

Syntax

```
Set-QADPSSnapinSettings [-DefaultExcludedProperties <String[]>]
[-DefaultPropertiesExcludedFromNonBaseSearch <String[]>]
[-Integer8AttributesThatContainDateTimes <String[]>]
[-Integer8AttributesThatContainNegativeTimeSpans <String[]>] [-DefaultPageSize <Int32>]
[-DefaultSizeLimit <Int32>] [-DefaultSearchScope <SearchScope>] [-DefaultWildcardMode
<WildcardMode>] [-DefaultOutputPropertiesForUserObject <String[]>]
[-DefaultOutputPropertiesForGroupObject <String[]>]
[-DefaultOutputPropertiesForComputerObject <String[]>]
[-DefaultOutputPropertiesForAdObject <String[]>]
[-DefaultOutputPropertiesForPasswordSettingsObject <String[]>]
```

Parameters

DefaultExcludedProperties

Use this parameter to specify the attributes that are excluded from processing by the UseDefaultExcludedProperties parameter on any particular cmdlet. Supply a list of the attribute LDAP display names as the parameter value.

DefaultOutputPropertiesForAdObject

Use this parameter to specify the default list of the object attributes that are to be retrieved from the directory and stored in the local memory cache by a Get- cmdlet during a search for objects other than a User, Group, Computer, or Password Settings object. Supply a list of the attribute LDAP display names as the parameter value.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by a Get- cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

DefaultOutputPropertiesForComputerObject

Use this parameter to specify the default list of the Computer object attributes that are to be retrieved from the directory and stored in the local memory cache by a Get- cmdlet during a search for Computer objects. Supply a list of the attribute LDAP display names as the parameter value.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by a Get- cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

DefaultOutputPropertiesForGroupObject

Use this parameter to specify the default list of the Group object attributes that are to be retrieved from the directory and stored in the local memory cache by a Get- cmdlet during a search for Group objects. Supply a list of the attribute LDAP display names as the parameter value.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by a Get- cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

DefaultOutputPropertiesForPasswordSettingsObject

Use this parameter to specify the default list of the Password Settings object attributes that are to be retrieved from the directory and stored in the local memory cache by a Get- cmdlet during a search for Password Settings objects. Supply a list of the attribute LDAP display names as the parameter value.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by a Get- cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

DefaultOutputPropertiesForUserObject

Use this parameter to specify the default list of the User object attributes that are to be retrieved from the directory and stored in the local memory cache by a Get- cmdlet during a search for User objects. Supply a list of the attribute LDAP display names as the parameter value.

Note: Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by a Get- cmdlet. If a particular attribute is not in the cache, the output object may not have a property that would provide access to the value of the attribute.

DefaultPageSize

Specify a new default value of the PageSize parameter for the Get- cmdlets. This page size value is used if the PageSize parameter is omitted. Initially, the default value of the PageSize parameter is set to 50.

DefaultPropertiesExcludedFromNonBaseSearch

Use this parameter to specify the attributes that are not to be retrieved from the directory and stored in the local memory cache by any particular Get- cmdlet during a search with the search scope other than 'Base'. Supply a list of the attribute LDAP display names as the parameter value.

Note: If a cmdlet does not cache a particular attribute, then the output object returned by the cmdlet may not have a property that would provide access to the value of the attribute.

DefaultSearchScope

Specify a new default value of the SearchScope parameter for the Get- cmdlets. Acceptable vales are:

- 'Base'
- 'OneLevel'
- 'Subtree'

This search scope setting is used if the SearchScope parameter is omitted. Initially, the default value of the SearchScope parameter is set to 'Subtree'.

DefaultSizeLimit

Specify a new default value of the SizeLimit parameter for the Get- cmdlets. This limitation on the size of the search result set is used if the SizeLimit parameter is omitted. Initially, the default value of the SizeLimit parameter is set to 1000.

DefaultWildcardMode

Specify a new default value of the WildcardMode parameter for the Get- cmdlets. Acceptable vales are:

- 'Ldap'
- 'PowerShell'

This wildcard mode setting is used if the WildcardMode parameter is omitted. Initially, the default value of the WildcardMode parameter is set to 'Ldap'.

Integer8AttributesThatContainDateTimes

Use this parameter to specify the Integer8 attributes whose values you want to be represented as DateTime objects in the output of the Get- cmdlets by default. Supply a list of the attribute LDAP display names as the parameter value.

Note: This setting applies only to the properties of a cmdlet's output object that have the member type of NoteProperty. Such properties are normally added to the output object in order to provide access to the attribute values of the respective directory object that are loaded to the local memory cache but cannot be accessed by using properties of the base object (the object for which the output object serves as a wrapper).

Integer8AttributesThatContainNegativeTimeSpans

Use this parameter to specify the Integer8 attributes whose values you want to be represented as TimeSpan objects in the output of the Get- cmdlets by default. Supply a list of the attribute LDAP display names as the parameter value.

The output TimeSpan objects represent the absolute values of the attributes, and thus indicate positive time intervals regardless of whether an actual attribute value is a negative or positive time interval.

Note: This setting applies only to the properties of a cmdlet's output object that have the member type of NoteProperty. Such properties are normally added to the output object in order to provide access to the attribute values of the respective directory object that are loaded to the local memory cache but cannot be accessed by using properties of the base object (the object for which the output object serves as a wrapper).

Detailed Description

You can use this cmdlet to modify some default settings that have effect within this PowerShell snap-in on any cmdlet where those settings are applicable. The changes you make to default settings are in effect during the current PowerShell session, and are discarded once you close the PowerShell console window. To view the default settings that are currently in effect, use the **Get-QADPSSnapinSettings** cmdlet.

Examples

Example 1

Configure the **Get-** cmdlets to return all search results by default (rather than limit the search result set to a maximum of 1000 items):

```
C:\PS> Set-QADPSSnapinSettings -DefaultSizeLimit 0
```

Example 2

Configure the Get- cmdlets to cache the 'msDS-ReplAttributeMetaData' attribute when retrieving User objects from the directory, in addition to the other attributes that are cached by default:

```
C:\PS> $list = Get-QADPSSnapinSettings –DefaultOutputPropertiesForUserObject
C:\PS> $list += 'msDS-ReplAttributeMetaData'
C:\PS> Set-QADPSSnapinSettings -DefaultOutputPropertiesForUserObject $list
```

Caching an attribute guarantees that the value of the attribute can be read by using properties of the output object returned by the Get- cmdlet. Thus, after you have changed configuration so as to cache the 'msDS-ReplAttributeMetaData' attribute, you can view the value of that attribute on a user account by using the following command:

```
C:\PS> Get-QADUser 'MyDomain\JSmith' | Format-Table name, 'msDS-ReplAttributeMetaData'
```