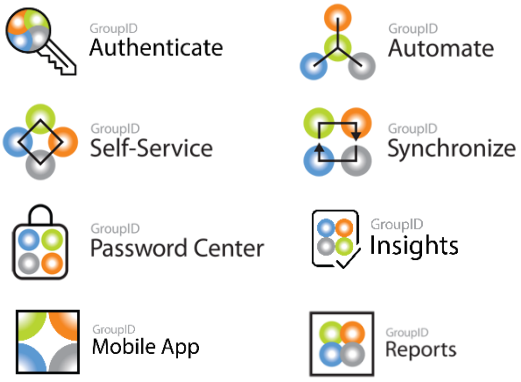
GroupID
by imanami

Version 10.0



GroupID Management Shell Command Reference

This publication applies to Imanami GroupID Version 10.0 and subsequent releases until otherwise indicated in new editions.

Contents

Chapter 1 - Introduction

GroupID Management Shell is a command-line interface for managing objects like users, contacts, mailboxes, groups, smart groups, dynasties and for performing other administrative tasks in an Active Directory and Azure based identity stores.

Built with Microsoft Windows PowerShell technology, GroupID Management Shell provides a platform to perform many of the tasks you can perform with GroupID Management Console as well as tasks that the console does not support.

This guide is a reference for the GroupID PowerShell commands. It provides detail on their function, syntax, parameters, and gives ready-to-use examples that you can modify and test in your own environment.

This guide is intended for advanced users familiar with the use of the Windows Command Prompt and Windows PowerShell.

To use GroupID Management Shell remotely, remoting feature needs to be enabled. See Connecting GroupID Management Shell Remotely on page 2 for details.

Identity Store based Model

GroupID 10.0 has extensible identity store based model. It supports the following data stores for creating an identity store:

- Active Directory
- Generic LDAP
- Microsoft Azure
- Digium Switchvox
- G Suite
- Health Meter

The commandlets cover in this guide are for Active Directory and Azure based identity stores.

Connecting GroupID Management Shell Remotely

To connect GroupID Management Shell remotely from another machine you need to configure the GroupID machine.

The logged-in user machine must be a member of the Administrators group on the remote GroupID machine or the logged-in user must be able to provide the credentials of an Administrator.

Make sure the following is available at the remote machine:

- Windows Powershell 2.0 or later
- Microsoft .Net Framework 4.7.2
- Windows Remote Management 2.0

To enable remoting on a GroupID machine with Windows Server 2008R2, Windows network location of that machine must be *Domain* or *Private* ("*Home*" or "*Work*"). If the network location is *Public*, GroupID Management Shell cannot create the required firewall exception for WS-Management Communication.

The Windows Management Shell remoting features are supported by the WS-Management protocol and the Windows Remote Management (WinRM) service that implements WS-Management in Windows.

1. Click **Start** > **Windows Powershell**.
   Right-click Windows PowerShell and select **Run as administrator**.
2. At the command prompt, type:
   enable-psremoting

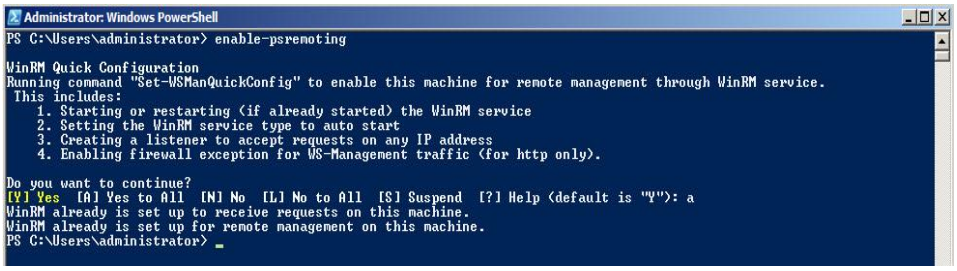


Figure 1: Windows PowerShell window

By default, on Windows Server® 2012, Windows PowerShell remoting is enabled. Use this command to re-enable remoting on Windows Server 2012 if it becomes disabled.

You have to run this command only one time on each computer that will receive commands. You do not have to run it on computers that

only send commands. Because the configuration starts listeners, it is prudent to run it only where it is needed.

To verify that remoting is configured correctly, run a test command:

```
new-PSSession –ComputerName <computer name>
```

This command creates a remote session on the local computer and return an object that represents the session. The output should look as shown in the following snapshot:

Figure 2: Windows PowerShell window – with New-PSSession command

Access GroupID Management Shell Remotely

Logon to the machine through which you want to remotely access the **GroupID Management Shell** and perform the following steps:

1. Right-click **Start** > **Imanami** > **GroupID Management Shell** using the **Run as Administrator** command to open it with Administrator privileges.
2. At the prompt, type the following script. It will display the new session created for the remote machine.

```
$username = "demo1\administrator"
$pass  = ConvertTo-SecureString "support123R" -
AsPlainText -Force
$Cred = New-Object
System.Management.Automation.PSCredential ($username,
$pass)
$scriptBlock = [Scriptblock]::Create("add-pssnapin -
Name Imanami.groups.management.powershell.admin10")
$s = New-PSSession -ComputerName "msvr02" -Credential
$Cred
Invoke-Command -Session $s -ScriptBlock $scriptBlock
Import-PSSession -Session $s -Type cmdlet
Connect-IdentityStore -mode "2" -IdentityStoreID "1" -
Credential $Cred
```

Following line of the above script connects the current user (*demo1\administrator*) to the identity store (*having ID 1*)

```
Connect-IdentityStore -mode "2" -IdentityStoreID "1" -
Credential $Cred
```


Using the following instructions, you can get the identity store ID from GroupID SQL database in which the desired identity store exists:

a. Login to SQL server (*having GroupID database*) with account having read permissions.

b. View the table "**Svc.Identitystore**" top 100 rows.

   See the following snapshot for details:

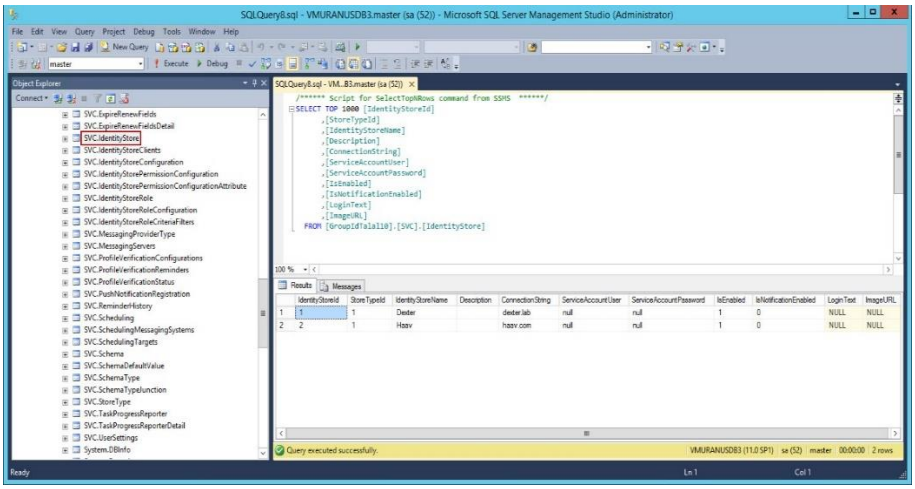


Figure 3: GroupID Database table in SQL Server for identity store ID

List of all commandlets

You can get the list of all GroupID Management Shell commandlets using the **Get-ImanamiCommand**.

1. Add-GroupMember
2. Clear-MessagingServer
3. Clear-Notifications
4. Clear-SmtpServer
5. Connect-IdentityStore
6. Convert-Group
7. ConvertTo-StaticGroup
8. Disable-DistributionGroup
9. Enable-DistributionGroup
10. Expire-Group

11. Extend-User
12. Get-AvailableMessagingServers
13. Get-Client
14. Get-Computer
15. Get-ConnectedStoreInformation
16. Get-ConnectedUser
17. Get-Contact
18. Get-Group
19. Get-GroupIdInformation
20. Get-GroupMember
21. Get-IdentityStore
22. Get-IdentityStoreRoles
23. Get-ImanamiCommand
24. Get-LogSettings
25. Get-MailBox
26. Get-Object
27. Get-Options
28. Get-ReplicationStatus
29. Get-RolePermissionNames
30. Get-Schedule
31. Get-SchemaAttributes
32. Get-SmartGroup
33. Get-SmsGateways
34. Get-Status
35. Get-TargetSchedule
36. Get-Token
37. Get-TombstoneObject
38. Get-User
39. Get-UserEnrollment

40. Get-UserRole
41. Invoke-Replication
42. Invoke-Schedule
43. Move-Group
44. New-Contact
45. New-Container
46. New-Dynasty
47. New-Group
48. New-IdentityStore
49. New-MailBox
50. New-Schedule
51. New-SmartGroup
52. New-User
53. Reinstate-User
54. Remove-Contact
55. Remove-Container
56. Remove-Group
57. Remove-GroupMember
58. Remove-IdentityStore
59. Remove-MailBox
60. Remove-Schedule
61. Remove-User
62. Renew-Group
63. Restore-TombstoneObject
64. Send-Notification
65. Send-TestNotification
66. Set-Contact
67. Set-Dynasty
68. Set-Group

69. Set-IdentityStore
70. Set-IdentityStoreRole
71. Set-MailBox
72. Set-MessagingServer
73. Set-Notifications
74. Set-Object
75. Set-Options
76. Set-Schedule
77. Set-SmartGroup
78. Set-SmtpSrever
79. Set-User
80. Stop-Schedule
81. Terminate-DirectReports
82. Transfer-DirectReports
83. Update-Group
84. Upgrade-Group

The following chapters provide information about syntax and supported parameters of these commandlets. Examples of each commandlet are also provided for further clarification.

Chapter 2 - Establishing Connection with Identity Store

This chapter covers commandlets for establishing connection with an identity store:

- Connect-IdentitySore: connects to an identity store using the authentication mode mentioned.
- Get-Token: gets a token from GroupID Security Service.

Review the description of the supported parameters of these commandlets along with their attributes and description in the List of Parameters table.

Common parameters of Windows Management Shell are not supported in GroupID Management Shell.

Connect-IdentityStore

If an identity store of the connected domain is available, then GroupID Management Shell gets connected to that identity store upon its launch. In case it does not exist the **Connect-IdentityStore** commandlet establishes a connection with the required identity store.

After a connection is established with the identity store you can then perform functions in directory as per your role and permissions.

Syntax

```
Connect-IdentityStore
  [-AuthenticationMode <string>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- None

Example

The following command connects you to the identity store specified by the **IdentityStoreId** parameter using the specified authentication mode and credentials that you set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A.

```
Connect-IdentityStore -AuthenticationMode 2 -
IdentityStoreId 2 -Credential $Cred
```

Get-Token

When Management Shell is connected to an identity store a token is passed with the commandlet enabling user to perform the required functions in directory.

If you want to perform a function in a different identity store Management Shell is connected with then first, you must have a valid token for the required identity store using the **Get-Token** commandlet. This commandlet gets a token from GroupID Security Service which was assigned to the user at the time of authentication.

Get-Token command is also used to get a valid token in case of token expires in a session.

Syntax

```
Get-Token
  [-AuthenticationMode <string>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- None

Example

The following command returns a token for the identity store specified by the **IdentityStoreId** parameter using the specified authentication mode and credentials that you set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A.

```
Get-Token -AuthenticationMode 2 -IdentityStoreId 2 -
Credential $Cred
```

Chapter 3 - General Commands

This chapter covers commandlets for performing general tasks such as:

- Get-Computer: provides information about computer object.
- Get-ConnectedStoreInformation: provides information about the connected identity store.
- Get-ConnectedUser: provides information about the connected user.
- Get-GroupIdInformation: provides information about GroupID.
- Get-ImanamiCommand: provides basic information about GroupID Management Shell commandlets.
- Get-ReplicationStatus: provides replication status of objects.
- Get-TombStoneObject: displays information about the tombstone objects.
- Invoke-Replication: starts replication process for all the identity stores or for a specific identity store.
- New-Container: creates a new organizational unit.
- Remove-Container: removes an empty organizational unit.
- Restore-TombSoneObject: restores tombstone objects from Directory.
- Send-Notification: sends notifications to a group or a particular user.

Review the description of the supported parameters of these commandlets along with their attributes in the List of Parameters table.

Common parameters of Windows Management Shell are not supported in GroupID Management Shell.

Get-Computer

The **Get-Computer** cmdlet retrieves the information about a computer object from the connected identity store. The computer can be a domain controller or an exchange server or just a simple client connected to the domain.

Syntax

```
Get-Computer
  [-Identity <string>]
  [<CommonParameters>]
```

Required parameter

- None

Example 1

This example retrieves a computer with a name arsalanahmadsvm.

```
get-computer -Identity arslanahmadsvm
```

Get-ConnectedStoreInformation

The **Get-ConnectedStoreInformation** commandlet retrieves information about the identity store connected to the current instance of the management shell.

Syntax

```
Get-ConnectedStoreInformation
  [<CommonParameters>]
```

Required parameter

- None

Example 1

The example displays name of the connected identity store, the last replication time to Elasticsearch, and messaging servers configured in the connected identity store.

```
Get-ConnectedStoreInformation
```

Get-ConnectedUser

Retrieves the general information about the user connected to the current instance of Management Shell.

Syntax

```
Get-ConnectedUser
  [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <PSCredential>]
  [-WarningAction <ActionPreference>]
  [-InformationAction <ActionPreference>]
  [-WarningVariable <String>]
  [-InformationVariable <String>]
  [-PipelineVariable <String>]
 [<CommonParameters>]
```

Required parameter

- None

Example 1

The example displays the logon name of the connected user, account locked information, identity store name, role name(s), and ObjectGuid.

```
Get-ConnectedUser
```

Get-GroupIdInformation

The **Get-GroupIdInformation** commandlet retrieves general information about GroupID.

Syntax

```
Get-GroupIdInformation
  [<CommonParameters>]
```

Required parameter

- None

Example 1

This example displays the name of the database and name of the SQL server being used by GroupID, GroupID version and the installation path of GroupID.

```
Get-GroupIdInformation
```

Get-ImanamiCommand

Use the **Get-ImanamiCommand** commandlet to retrieve basic information about GroupID Management Shell commandlets and other command elements.

Syntax

```
Get-ImanamiCommand
  [-Name <string[]>]
  [-Verb <string>]
  [-Noun <string>]
  [-AttributesToLoad <string[]>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- None

Example 1

The following command shows information about all commandlets.

```
Get-ImanamiCommand
```

Example 2

The following command gets all commandlets and command elements with the word **Set** in their name.

```
Get-ImanamiCommand -Name Set*
```

Example 3

The following command gets all commandlets and command elements with the letter **Y** anywhere in the verb of their name.

```
Get-ImanamiCommand -Verb *Y*
```

Get-ReplicationStatus

The **Get-ReplicationStatus** commandlet retrieves the replication status of each domain of an identity store. The commandlet provides replication status of each object (such as users, groups, contact, computer, public folders and OUs) in the identity store domain(s).

Syntax

```
Get-ReplicationStatus
  [-IdentityStoreName] <string>
  [<CommonParameters>]
```

Required parameter

- IdentityStoreName

Example 1

The following commandlet provides date and time information when the objects of an identity store are replicated to Elasticsearch and the time elapsed since last replication.

```
Get-ReplicationStatus -IdentityStoreName AdStore8
```

Get-TombStoneObject

When you delete an object from Directory, the object is not physically removed from the database. Instead, Directory marks the object as deleted, strips most of the properties from the object and moves it to a special container. The object becomes invisible to normal directory operations and is referred to as a **tombstone** object. Use the Get-TomStoneObject commandlet to view information about these tombstone objects.

Syntax

```
Get-TombstoneObject
  [[-Identity] <string[]>]
  [-SearchContainer <string[]>]
  [-SearchContainersScopeList <string>]
  [-ShouldReturnCollection]
  [-MaxItemsToDisplay <int>]
  [-ObjectType <string[]>]
  [-LdapFilter <string>]
  [-SmartFilter <string>]
  [-ServerFilter <string>]
  [-AttributesToLoad <string[]>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- None

Example 1

The following command retrieves all tombstone objects from Directory, using the credentials of current user logged-on to the identity store.

```
Get-TombStoneObject
```

Example 2

The following command retrieves the tombstone group **Event Management**, using the credentials set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A.

```
Get-TombStoneObject -identity "Event Management" -
Credential $Cred
```

Example 3

The following command retrieves all tombstone objects with display names starting with the letter **S**.

```
Get-TombStoneObject -LdapFilter "(CN = S*)"
```

Invoke-Replication

This will start replication process for all the identity stores or specific identity store.

Syntax

```
Invoke-Replication
  [-IdentityStoreId <int>]
  [-DeletedObjects]
  [-RestoreReplication]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- None

Example 1

The following command replicate identity store with ID 1.

```
Invoke-Replication -IdentitystoreId 1
```

Example 2

The following command replicate deleted objects for identity store with ID 1.


```
Invoke-Replication -IdentitystoreId 1 -DeletedObjects
```

Example 3

The following command will start restoration of replication for identity store with ID 1.

```
Invoke-Replication -IdentitystoreId 1 -RestoreReplication
```

New-Container

The **New-Container** commandlet creates a new organizational unit in Directory. You can also use it to create nested organizational units by repeatedly executing the commandlet and changing the value of the **ParentContainer** parameter.

Syntax

```
New-Container
  -ContainerName <string[]>
  -OrganizationalUnit <string>
  [-AccidentalDeletion]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameters

- ContainerName
- OrganizationalUnit

Example 1

The following command creates the organizational unit **Recruiting** at the root level in Directory, using the credentials of current user logged-on to the identity store.

```
New-Container -OrganizationalUnit "DC=HR,DC=Imanami,DC=US"
-ContainerName "Recruiting"
```

Example 2

The following command creates the organizational unit **Local Recruiting** inside the **Recruiting** container in Directory using the credentials set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A

```
New-Container - OrganizationalUnit
"OU=Recruiting,DC=HR,DC=Imanami,DC=US" -ContainerName
"Local Recruiting" -Credential $Cred
```

Remove-Container

Use the **Remove-Container** commandlet to delete organizational units from Directory. The commandlet only supports deletion of containers at leaf level, having no objects. If the container contains objects or sub-containers, the commandlet does not process the request and throws an exception.

Syntax

```
Remove-Container
  -Identity <string>
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example 1

The following command removes the **Miscellaneous** container, using the credentials of current user logged-on to the identity store.

```
Remove-Container -identity
"OU=Miscellaneous,OU=Recruiting,DC=HR,DC=Imanami,DC=US"
```

Example 2

The following command first shows the changes that result from executing the command. The command uses the credentials set in the **$Credentials** environment variable to perform the deletion. For information about setting credentials, see Appendix A.

```
Remove-Container -identity
"OU=Miscellaneous,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -
Credential $Cred
```

Restore-TombStoneObject

Use the **Restore-TombStoneObject** commandlet to restore tombstone objects from Directory. The tombstone object is restored as an unmanaged group with all supported attributes to its original container. If the parent container has been deleted, the commandlet also reinstates the container for the group.

Syntax

```
Restore-TombstoneObject
  [-Identity] <string>
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- None

Example

The following command restores the tombstone group **Event Management**, using the credentials set in the **$Creds** environment variable. For information about setting credentials, see Appendix A.

```
Restore-TombStoneObject –identity "Event Management" –
Credential $Cred
```

Send-Notification

Use the **Send-Notification** commandlet to send notifications to a group or a particular user. GroupID modules automatically generate e-mail notifications upon the occurrence of certain events; for example, expiry of groups, execution of a job, and generation of workflow requests. The modules use template files for generating the contents of the notification e-mails. These template files are located at:

X:\Program Files\Imanami\GroupID 8.0\Automate\Templates\Notifications

Where X is the drive where the GroupID installation directory resides. The **Send-Notification** commandlet also requires a template file for generating e-mail contents. You can utilize one from the available templates or create your own.

The commandlet also requires an **SMTP server** and a **From** e-mail address that you can configure using the **Set-Options** commandlet.

Syntax

```
Send-Notification
  -Identity <string>
  -Subject <string>
  -TemplateFile <string>
  [-InlineImageFile <string>]
  [-QueueEmail]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameters

- Identity
- Subject
- TemplateFile

Example 1

The following commands first configure the **SMTP Server**, then set a **From** e-mail address, and finally send a group expiry notification to **UserA** using the credentials of current user logged-on to the identity store.

```
Set-Options -SmtpServer "HR.Imanami.US"
Set-Options -FromAddress "Administrator@HR.Imanami.US"

Send-Notification -Identity
"CN=UserA,CN=Users,DC=HR,DC=Imanami,DC=US" -Subject "Expiry
Notification" -TemplateFile "C:\Program
Files\Imanami\GroupID
8.0\Automate\Templates\Notifications\ExpiringTemplate.html"
-QueueEmail
```

Example 2

The following command sends a notification to the **New Arrivals** group. It follows a custom template with an in-line image and uses the credentials of the user set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A.

```
Send-Notification -Identity "CN=New
Arrivals,CN=Users,DC=HR,DC=Imanami,DC=US" -Subject "Welcome
to Imanami" -TemplateFile "C:\Welcome.html" -
InlineImageFile "C:\WelcomeNote.jpg" -QueueEmail
```

Chapter 4 - Identity Store Commands

This chapter covers commandlets for performing identity store related tasks such as:

- Clear-MessagingServer: removes a configured messaging server.
- Clear-Notifications: removes notification settings of an identity store.
- Clear-SmtpServer: removes a configured SMTP server of an identity store.
- Get-AvailableMessagingServers: retrieves messaging servers for the configured messaging provider.
- Get-Client: lists information about the GroupID clients.
- Get-IdentityStore: retrieves information about an identity store.
- Get-IdentityStoreRoles: retrieves information about the security roles of an identity store
- Get-LogSettings: provides information about the global log settings of the connected identity store.
- Get-RolePermissionNames: lists the names of the permissions assigned to the logged-in user.
- Get-SchemaAttributes: lists schema attributes available for an identity store.
- Get-SmsGateways: provides information of the configured SMS gateways.
- Get-UserRole: displays role information of the specified user of an identity store.
- New-IdentityStore: creates a new identity store.
- Remove-IdentityStore: removes an identity store in GroupID.
- Send-TestNotification: sends a test notification.
- Set-IdentityStore: modifies an identity store configuration.
- Set-IdentityStoreRole: modifies properties of a security role in an identity store.
- Set-MessagingServer: configures a messaging server in identity store.
- Set-Notifications: modifies notification settings of an identity store.
- Set-SmtpServer: configures an SMTP server of an identity store.

Clear-MessagingServer

The commandlet **Clear-MessagingServer** removes the configured messaging server from the specified identity store.

This cmdlet will also clear the SMTP settings, notification settings, password expiry settings, membership lifecycle notification settings, and managed by notification settings for the identity store.

Syntax

```
Clear-MessagingServer
  -IdentityStoreName <string>
  [<CommonParameters>]
```

Required parameters

- IdentityStoreName

Example 1:

This example clears configured messaging server for *AdStore8* identity store.

```
Clear-MessagingServer -IdentityStoreName AdStore8
```

Clear-Notifications

The commandlet **Clear-Notifications** removes notifications settings from an identity store. The notifications settings can be removed individually or in sets.

Syntax

```
Clear-Notifications
  -IdentityStoreName <string>
  [-PrimaryRecepients]
  [-CarbonCopy]
  [-NotifyLoggedInUsers]
  [-NotifyOwners]
  [-NotifyModifiedObject]
  [-NotifyPublicGroupOwner]
  [-NotifyAddedMembers]
  [-PasswordPortalUrl]
  [-NotifyUserGroupJoinML]
  [-NotifyUserGroupLeaveML]
  [-XDaysBeforeLeaveNotificationML]
  [-NotifyUserGroupJoinMB]
  [-NotifyUserGroupLeaveMB]
```

```
  [-XDaysBeforeLeaveNotificationMB]
  [<CommonParameters>]
```

```
Clear-Notifications
  -IdentityStoreName <string>
  [-ClearSet {All | Recipients | PasswordExpiry | ML | MB}]
  [<CommonParameters>]
```

Required parameter

- IdentityStoreName

Example 1

This example individually removes the Membership Lifecycle notification option – *X days before user is going to leave the group* for the *AdStore9* identity store.

```
Clear-Notifications -IdentityStoreName AdStore9 -
NotifyLoggedInUsers -XdaysBeforeLeaveNotificationML 10
```

Example 2

This example removes recipients in sets mentioned under the Recipients section on the Notification page of *AdStore9* identity store properties.

```
Clear-Notifications -IdentityStoreName AdStore9 -ClearSet
Recipients
```

Clear-SmtpServer

The commandlet **Clear-SmtpServer** removes the SMTP server configurations from an identity store.

This cmdlet will also clear the notification settings for the identity store recipients, password expiry group notifications, membership lifecycle notifications, and managed by notification options for the specified identity store.

Syntax

```
Clear-SmtpServer
  -IdentityStoreName <string>
  [<CommonParameters>]
```

Required parameter

- IdentityStoreName

Example 1

This example clears the configured SMTP server in *AdStore9* identity store.

```
Clear-SmtpServer -IdentityStoreName AdStore9
```

Get-AvailableMessagingServers

The commandlet **Get-AvailableMessagingServers** retrieves the messaging server(s) available for the configured messaging provider.

Syntax

```
Get-AvailableMessagingServers
  -IdentityStoreName <string>
  -Provider {o365 | gsuite | exchange2010 | exchange2013 |
exchange2016 | exchange2019}
  -UserName <string>
  [-Password <string>]
  <CommonParameters>]
```

```
Get-AvailableMessagingServers
  -IdentityStoreName <string>
  -Provider {o365 | gsuite | exchange2010 | exchange2013 |
exchange2016 | exchange2019}
  -Credential <pscredential>
  [<CommonParameters>]
```

Required parameter

- IdentityStoreName
- Provider
- Credential

Example 1

This example retrieves the available messaging server(s) configured in *AdStore1* identity store for Exchange 2010 messaging provider.

```
Get-AvailableMessagingServers -IdentityStoreName AdStore1 -
Provider exchange2010 -UserName administrator -Password
webdir123R -Domain pucit.local
```

Example 2

This example retrieves the available messaging server(s) configured in *Adstore1* identity store for Office365 messaging provider.

```
Get-AvailableMessagingServers -IdentityStoreName AdStore1 -
Provider o365 -UserName admin@mydomain.onmicrosoft.com -
Password webdir123R -Domain mydomain.onmicrosoft.com -AppId
'eeeeeeee-aaaa-dddd-bbbb-ccccccccccc'
```

Example 3

This example retrieves the available messaging server(s) in *AdStore1* identity store for GSuite (Google Apps) messaging provider.

```
Get-AvailableMessagingServers -IdentityStoreName AdStore1 -
Provider gsuite -UserName svcaccount@myproject-
219211.iam.gserviceaccount.com -AdminUsername
'arslan@mydomain.com' -P12CertificatePath
'C:\Keys\gsuite\key.p12'
```

Get-Client

The commandlet **Get-Client** gets information about the GroupID clients such as Automate, Management Shell, GroupID Mobile Service, each Self-Service portal, each Password Center portal. The information includes client name, client type, and identity store(s) of the client.

Syntax

```
Get-Client
  [[-ClientName] <String>]
  [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <PSCredential>]
  [-WarningAction <ActionPreference>]
  [-InformationAction <ActionPreference>]
  [-WarningVariable <String>]
  [-InformationVariable <String>]
  [-PipelineVariable <String>]
 [<CommonParameters>]
```

Required parameter

- None

Example 1

This example retrieves information about a client *Automate ARSALANAHMADVM*.

```
Get-Client –ClientName ‘Automate ARSLANAHMADVM’
```


Example 2

This example retrieves information about two clients – *automate arslanahmadvm* and *managementshell arsalanahmadvm* – through the pipeline operator.

```
'automate arslanahmadvm', 'managementshell arslanahmadvm' |
Get-Client
```

Example 3

This example lists all GroupID clients available on the GroupID machine.

```
Get-Client
```

Example 4

This example lists all Password Center clients.

```
Get-Client | Where-Object {$_.Type -eq "Password Center"}
```

Example 5

This example lists all GroupID clients that belong to identity store *AdStore9*.

```
Get-Client | Where-Object
{$_.IdentityStores.Contains('AdStore9')}
```

Get-IdentityStore

The commandlet **Get-IdentityStore** retrieves information about the specified identity store or retrieves information of identity store(s) as per the given switches such as *All*, *Connected*, *Enabled* or *Disabled*.

The information includes identity store name, description, connection string, notification status, roles in identity store, and so on.

Syntax

```
Get-IdentityStore
  -IdentityStoreName <String>
  [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <PSCredential>]
  [-WarningAction <ActionPreference>]
  [-InformationAction <ActionPreference>]
  [-WarningVariable <String>]
  [-InformationVariable <String>]
  [-PipelineVariable <String>]
  [<CommonParameters>]
```

```
Get-IdentityStore
  -All
  [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <PSCredential>]
  [-WarningAction <ActionPreference>]
  [-InformationAction <ActionPreference>]
  [-WarningVariable <String>]
  [-InformationVariable <String>]
  [-PipelineVariable <String>]
 [<CommonParameters>]
```

```
Get-IdentityStore
  -Connected
  [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <PSCredential>]
  [-WarningAction <ActionPreference>]
  [-InformationAction <ActionPreference>]
  [-WarningVariable <String>]
  [-InformationVariable <String>]
  [-PipelineVariable <String>]
  [<CommonParameters>]
```

```
Get-IdentityStore
  -Enabled
  [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <PSCredential>]
  [-WarningAction <ActionPreference>]
  [-InformationAction <ActionPreference>]
  [-WarningVariable <String>]
  [-InformationVariable <String>]
  [-PipelineVariable <String>]
  [<CommonParameters>]
```

```
Get-IdentityStore
  -Disabled
  [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <PSCredential>]
  [-WarningAction <ActionPreference>]
  [-InformationAction <ActionPreference>]
  [-WarningVariable <String>]
  [-InformationVariable <String>]
  [-PipelineVariable <String>]
  [<CommonParameters>]
```


Required parameter

- IdentityStoreName or a switch { All | Connected | Enabled | Disabled}

Example 1

This example retrieves information of *AdStore1* identity store.

```
Get-IdentityStore -IdentityStoreName AdStore1
```

Example 2

This example retrieves information of two identity stores – *AdStore1* and *AdStore2* – through the pipeline operator.

```
'AdStore1','AdStore2' | Get-IdentityStore
```

Example 3

This example retrieves information of all identity stores available on the GroupID machine.

```
Get-IdentityStore -All
```

Example 4

This example retrieves information of identity store connected to the current instance of the GroupID Management Shell.

```
Get-IdentityStore -Connected
```

Example 5

This example displays information of all enabled identity store(s).

```
Get-IdentityStore -Enabled
```

Example 6

This example is for getting information of all disabled identity store(s).

```
Get-IdentityStore -Disabled
```

Get-IdentityStoreRoles

The commandlet **Get-IdentityStoreRoles** retrieves information about the security roles associated with an identity store. The information includes role name, role priority, role criteria and role permissions.

Syntax

```
Get-IdentityStoreRoles
  [-IdentityStoreName] <String> [[-RoleName] <String>]
  [[-Subset] <String>]
  [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <PSCredential>]
  [-WarningAction <ActionPreference>]
  [-InformationAction <ActionPreference>]
  [-WarningVariable <String>]
  [-InformationVariable <String>]
  [-PipelineVariable <String>]
  [<CommonParameters>]
```

Required parameter

- IdentityStoreName

Example 1:

This example retrieves information of *customrole1* role in *AdStore1* identity store.

```
Get-IdentityStoreRoles -IdentityStoreName AdStore1 -
RoleName customrole1
```

Example 2:

This example provides information about all roles in *adstore1* identity store.

```
Get-IdentityStoreRoles -IdentityStoreName adstore1
```

Example 3:

This example retrieves information about two security roles – *customrole1* and *customrole2* – in *AdStore1* identity store through the pipeline operator.

```
'customrole1', 'customrole2' | Get-IdentityStoreRoles -
IdentityStoreName AdStore1
```

Get-LogSettings

The commandlet **Get-LogSettings** provides information about the global log settings of the identity store connected with this instance of GroupID Management Shell.

Syntax

```
Get-LogSettings
  [<CommonParameters>]
```

Required parameter

- None

Example 1

This example retrieves the log settings of the connected identity store.

```
Get-LogSettings
```

Get-RolePermissionNames

The commandlet **Get-RolePermissionNames** helps user to see the names of the permissions that can be assigned to / revoked from a security role in an identity store.

Syntax

```
Get-RolePermissionNames
  [-IncludeEntityTypes]
  [<CommonParameters>]
```

Required parameter

- None

Example 1

This example provides list of permissions names for a security role.

```
Get-RolePermissionNames
```

Example 2

This example provides list of permission names along with the category of a permission.

```
Get-RolePermissionNames -IncludeEntityTypes
```

Get-SchemaAttributes

The commandlet **Get-SchemaAttribute** enables you to retrieve comprehensive list of schema attributes available for an identity store.

This cmdlet can be used to enlist the names of schema attributes required for various cmdlets like cmdlets related to identity store roles etc.

Syntax

```
Get-SchemaAttributes
  [-IdentityStoreName] <string>
  [<CommonParameters>]
```

Required parameter

- IdentityStoreName

Example 1

This example retrieves list of available schema attributes in alphabetical order for the *AdStore9* identity store.

```
Get-SchemaAttributes -IdentityStoreName AdStore9
```

Get-SmsGateways

The commandlet **Get-SmsGateways** provides information of the SMS gateways configured in GroupID.

Syntax

```
Get-SmsGateways
  [<CommonParameters>]
```

Required parameter

- None

Example 1

This example lists all the configured SMS gateways in GroupID.

```
Get-SmsGateways
```

Get-UserRole

The commandlet **Get-UserRole** displays information about the role of the specified user in an identity store.

If a user has different roles in different GroupID clients of an identity store; and ClientName parameter is not specified, this commandlet displays the highest priority role of the user. If the identity store name is not specified, the connected identity store is used by this cmdlet.

Syntax

```
Get-UserRole
  [-Identity] <string>
  [-IdentityStoreName <string>]
  [-ClientName <string>]
  [-All]
  [<CommonParameters>]
```

Required parameter

- Identity

Example 1

This example provides role information of the user *testingaccount* in the *Automate ArslanAhmadVM* client of the *AdStore1* identity store.

```
Get-UserRole -Identity testingaccount -IdentityStoreName
AdStore1 -ClientName 'Automate ArslanAhmadVM'
```

Example 2

This example displays the highest priority role information of the *testingaccount*@pucit.*local* user for all clients of *AdStore1* identity store.

```
Get-UserRole -Identity testingaccount@pucit.local -
IdentityStoreName AdStore1
```

Example 3

This example displays role information of the *testingaccount*@pucit.*local* user in the *managementshell arsalanahmadvm* GroupID client of the connected identity store.

```
Get-UserRole -Identity testingaccount@pucit.local -
ClientName 'managementshell arslanahmadvm'
```

Example 4

This example retrieves the highest priority role of *testingaccount* user in the connected identity store. If the user has different roles in different GroupID clients, only the role having the highest priority is retrieved.

```
Get-UserRole -Identity testingaccount
```

Example 5

This example retrieves information of all roles of *testingaccount* user in all client of the connected identity store.

```
Get-UserRole -Identity testingaccount -All
```

Example 6

This example retrieves all roles of *euser1* and *euser2* users in the connected identity store through pipelining.

```
'euser1', 'euser2', 'testingaccount' | Get-UserRole -All
```

New-IdentityStore

The commandlet **New-IdentityStore** creates a new identity store. This commandlet requires valid credentials and connectivity before it creates the store. However, this behavior can be overridden by specifying the IgnoreConnectionFail parameter.

This cmdlet uses dynamic parameters based on the value of IdentityStoreType parameter. The parameters that become available depending on the values of IdentityStoreType are as follows:

- IdentityStoreType: ActiveDirectory
  Domain: The connection string / domain of the active directory.
- IdentityStoreType: WindowsAzure
  Domain: The domain of the Azure / Office365 store.
  AppId: The name of GroupID application registered in the Windows Azure admin panel.
- IdentityStoreType: GSuite
  AdminUsername: The username of the administrator of the GSuite account.
  P12CertificatePath: The path where the certificate file (.p12 extension) downloaded from Google Admin Console is placed (including the filename).

Syntax

```
New-IdentityStore
  -IdentityStoreType <IdentityStoreType>
  -IdentityStoreName <String>
  -Credential <PSCredential>
  [-Description <String>]
  [-PassThru]
  [-IgnoreConnectionFail]
  [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-WarningAction <ActionPreference>]
  [-InformationAction ActionPreference>]
  [-WarningVariable <String>]
  [-InformationVariable <String>]
  [-PipelineVariable <String>]
 [<CommonParameters>]
```

Required parameter

- IdentityStoreType
- IdentityStoreName
- Credential

Example 1

This example create a new Active Directory based identity store by explicitly specifying the credentials for the new identity store.

```
New-IdentityStore -IdentityStoreType ActiveDirectory -
IdentityStoreName DemoAdStore -UserName administrator -
Password webdir123R -Domain pucit.local
```

For an Active Directory based identity store, Domain parameter is mandatory.

Example 2

This example creates a new Active Directory based identity store by providing the secure credentials. Here $cred is an object of type PSCredential which was created by Get-Credential commandlet.

```
New-IdentityStore -IdentityStoreType ActiveDirectory -
IdentityStoreName DemoAdStore2 -Credential $cred -Domain
pucit.local
```

Example 2

This example creates an Azure based identity store.

```
New-IdentityStore -IdentityStoreType WindowsAzure -
IdentityStoreName DemoAzStore1 -UserName
admin@mydomain.onmicrosoft.com -Password webdir123R -Domain
mydomain.onmicrosoft.com -AppId 'aaaaaaaa-bbbb-cccc-dddd-
eeeeeeeeeeee'
```

In case of an Azure based identity store, Domain and AppId parameters are mandatory.

Example 3

This example creates a Google Apps (G-Suite) based identity store.

```
New-IdentityStore -IdentityStoreType GSuite -
IdentityStoreName DemoGStore1 -UserName svcacc@myproject-
111222.iam.gserviceaccount.com -AdminUsername
admin@mydomain.com -P12CertificatePath
'C:\Keys\gsuite\key.p12'
```

For Google Apps based identity store, AdminUserName and P12CertificatePath parameters are mandatory. However, 'Password' parameter is ignored.

Example 4

This example creates Google Apps (G-Suite) based identity store using secure credentials.

The $creds (an object of type PSCredential) object must contain the service account as username. The 'Password' property of this object can be anything but not empty.

```
New-IdentityStore -IdentityStoreType GSuite -
IdentityStoreName DemoGStore2 -Credential $creds -
AdminUsername arslan@bibelotz.com -P12CertificatePath
'C:\Keys\gsuite\key.p12'
```

Example 5

This example creates an Active Directory based identity store by ignoring the credential and connection details.

```
New-IdentityStore -IdentityStoreType ActiveDirectory -
IdentityStoreName DemoAdStore3 -UserName nouser -Password
wrongpwd -Domain nodomain.local -IgnoreConnectionFail
```

Remove-IdentityStore

The commandlet **Remove-IdentityStore** removes an identity store from GroupID.

Syntax

```
Remove-IdentityStore
  [-IdentityStoreName] <String>
  [-PassThru]
  [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <PSCredential>]
  [-WarningAction <ActionPreference>]
  [-InformationAction <ActionPreference>]
  [-WarningVariable <String>]
  [-InformationVariable <String>]
  [-PipelineVariable <String>]
  [<CommonParameters>]
```

Required parameter

- IdentityStoreName

Example 1

This example removes an identity store named *DemoAzStore1*.

```
Remove-IdentityStore -IdentityStoreName DemoAzStore1
```

Example 2

This example removes *DemoGStore1* and *DemoGStore2* identity stores through the pipeline operator.

```
'DemoGStore1', 'DemoGStore2' | Remove-IdentityStore
```

Send-TestNotification

The commandlet **Send-TestNotification** sends a test notification using the email addresses (specified From/To) through the SMTP server of the specified identity store. This cmdlet can be used to validate SMTP settings before configuring notifications or SMTP settings.

Syntax

```
Send-TestNotification
  -IdentityStorename <string>
  -SmtpServer <string>
  -FromEmail <string>
  -ToEmail <string>
  -Port <int>
  [-Credential <pscredential>]
  [-UseSmptUserAuthentication]
  [-SslEnabled]
  [<CommonParameters>]
```

Required parameter

- IdentityStoreName
- SmtpSesrver
- FromEmail
- Toemail
- Port

Example 1

This example sends a test notification to *euser1@pucit.local* using the SMTP server configured on port 25 for user *arsalanahmadsvm* in *AdStore1* identity store.

```
Send-TestNotification -IdentityStorename AdStore9 -
SmtpServer arslanahmadsvm.pucit.local -Port 25 -FromEmail
noreply@pucit.local -ToEmail euser1@pucit.local
```

Set-IdentityStore

The commandlet **Set-IdentityStore** modifies the identity store settings and configurations.

Many parameters of this cmdlet require the user to specify schema attribute names. You can use **Get-SchemaAttributes** commandlet to retrieve a list of attributes available for an identity store.

Syntax

```
Set-IdentityStore
  -IdentityStoreName <string>
  -Credential <pscredential>
  [-NewName <string>]
  [-StoreDescription <string>]
  [-StoreEnabled <bool>]
  [-RoleOperation {add | remove | remove all}]
  [-RoleName <string>]
  [-RoleDescription <string>]
  [-RolePriority <string>]
  [-RoleCriteriaScope {Group | Container}]
  [-RoleCriteriaDN <string>]
  [-RoleCriteriaOperator {Or | And}]
  [-RoleCriteriaFilters <string[][]>]
  [-RolePermissions <string[]>]
  [-RoleNameToCopy <string>]
  [-DefaultAllowRolePermissions]
  [-RoleReadonly]
  [-RoleSystemOnly]
  [-RoleDisabled]
  [-GroupExpiryQuantity <string>]
  [-GroupExpiryUnit {Never | Days | Weeks | Months | Years
| Indefinite}]
  [-GlmContainersPolicy {Exclude | Include}]
  [-GlmContainers <string[]>]
  [-GlmContainersOperation {add | remove | remove all}]
  [-EnableSecurityGroupsExpiry]
  [-DisableSecurityGroupsExpiry]
  [-EnableExpiredGroupDeletion]
  [-ExpiredGroupsDeletionInterval <string>]
```

```
  [-DisableExpiredGroupDeletion]
  [-EnableGUSLifecycle]
  [-GroupExtensionPolicy {Extend | Reduce}]
  [-GroupLifeDays <string>]
  [-DisableGUSLifecycle]
  [-EnableGroupAttestation]
  [-DisableGroupAttestation]
  [-DefaultApprover <string>]
  [-GlmNotifyOwnersXDaysBeforeOperation {add | remove |
remove all}]
  [-GlmNotifyOwnersXDaysBefore <string[]>]
  [-GlmEnableNotificationOfTodaysExpiry]
  [-GlmDisableNotificationOfTodaysExpiry]
  [-PrefixOperation {add | remove | remove all}]
  [-Prefixes <string[]>]
  [-HistoryTrackingOption {Nothing | All_Actions |
Selected_Actions}]
  [-HistoryActionsOperation {add | remove | remove all}]
  [-HistorySelectedActions {OwnershipChange |
AdditionalOwnerChange | ExpirationPolicyChange |
GroupExpireRenew | QueryChange | SecurityTypeChange |
ObjectCreated | ObjectDeleted | IdentityStoreHistory |
SecurityRolesHistory | WorkflowsHistory}]
  [-HistoryRetention {All | Last_30_Days | Last_60_Days |
Last_90_Days | Last_120_Days | Last_6_Months | Last_1_Year
| Last_2_Years | Last_5_Years}]
  [-FileLoggingEvent {All | Debug | Info | Warn | Error |
Off}]
  [-WindowsLoggingEvent {FailureAudit | SuccessAudit | Info
| Warn | Error}]
  [-MaximumMembersPerGroup <string>]
  [-WhenGroupMembershipThresholdReach {PreventUpdation |
NestIntoChildGroups}]
  [-EnableOrphanGroupsDeletion]
  [-DisableOrphanGroupsDeletion]
  [-EnableOutOfBoundsAlerts]
  [-DisableOutOfBoundsAlerts]
  [-MembershipCountThreshold <string>]
  [-MembershipPercentageThreshold <string>]
  [-ProfileValidationGroupDN <string>]
  [-RegularProfileValidationLifecycle <string>]
  [-EnableNewProfileValidationLifecycle]
  [-DisableNewProfileValidationLifecycle]
  [-NewProfileValidationLifecycle <string>]
  [-ProfileValidationReminderOperation {add | remove}]
  [-ProfileValidationReminders <string[][]>]
  [-ProfileValidationExtensionPeriod <string>]
  [-EnableAttributeUpdation]
  [-DisableAttributeUpdation]
  [-ProfileValidationAttributeName <string>]
  [-ProfileValidationAttributeValue <string>]
  [-EnableValidationDateRemoval]
```

```
  [-DisableValidationDateRemoval]
  [-ValidationDateRemovalInterval <string>]
  [-EnrollmentEnabled <bool>]
  [-AuthenticationTypeOperation {enable | disbale}]
  [-AuthenticationType <string[]>]
  [-QuestionOperation {add | remove | remove all}]
  [-SecurityQuestions <string[]>]
  [-PasswordExceptionOperation {add | remove | remove all}]
  [-PasswordExceptions <string[][]>]
  [-PasswordRuleOperation {add | remove | remove all}]
  [-PasswordRules <string[]>]
  [-DisallowingPasswordExceptionFilePath <string>]
  [-EnableSWAuthenticationViaSecurityQuestions]
  [-DisableSWAuthenticationViaSecurityQuestions]
  [-SWAQuestionsOperation {add | remove}]
  [-SWAQuestions <string[][]>]
  [-EnableSWAuthenticationViaMobile]
  [-DisableSWAuthenticationViaMobile]
  [-SWAMobileAttribute <string>]
  [- EnableSWAuthetnicationViaEmail]
  [-DisableSWAuthenticationViaEmail]
  [-SWEmailAttribute <string>]
  [-SWAuthenticationFactor <string>]
  [<CommonParameters>]
```

You can use the Set-IdentityStore commandlet in a secure way by using the *Credential* parameter or by specifying the credentials through *Username* and *Password* parameters in plain text format which is not a secure way.

Required parameter

- IdentityStoreName
- Credential / Username

Example 1

This example changes name of *AdStore9* identity store to *AdStore9_renamed*.

```
Set-IdentityStore -IdentityStoreName AdStore9 -NewName
'AdStore9_renamed' -Credential $creds -Domain pucit.local
```

Example 2

This example enables the *Email verification* authentication type for the *AdStore9* identity store.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -AuthenticationTypeOperation
enable -AuthenticationType 'Email Verification'
```

Example 3

This example disables enrollment for the *AdSore9* identity store.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -EnrollmentEnabled $false
```

Example 4

This example modifies the group lifecycle expiry policy of the *AdStore9* identity store to 21 days.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -GroupExpiryQuantity 21
```

Example 5

This example modifies the group lifecycle expiration policy of *AdStore9* identity store to 10 months.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -GroupExpiryQuantity 10 -
GroupExpiryUnit Months
```

Example 6

This example sets the group lifecycle expiration policy of the AdStore9 identity store to 'never' by setting value of the GroupExpiryUnit parameter to Indefinite. Even though the GroupExpiryQuantity parameter is set to any value.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -GroupExpiryQuantity 10 -
GroupExpiryUnit Indefinite
```

Example 7

This example Configures containers policy and add containers.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -GlmContainersPolicy Include -
GlmContainersOperation add -GlmContainers
'OU=WorkingOU,DC=pucit,DC=local','OU=ArslanAhmadOU,OU=Worki
ngOU,DC=pucit,DC=local'
```

Example 8

This example enables expiry of security groups, deletion of expired groups and sets interval of group deletion to 45 days.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -EnableSecurityGroupsExpiry -
EnableExpiredGroupDeletion -ExpiredGroupsDeletionInterval
45
```

Example 9

This example enables GUS lifecycle and reduces group's life if not used within 25 days.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -EnableGUSLifecycle -
GroupExtensionPolicy Reduce -GroupLifeDays 25
```

Example 10

This example enables group attestation feature and sets the *TestingAccoun@pucit.local* user as the default approver for the *AdStore9* identity store.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -EnableGroupAttestation -
DefaultApprover 'CN=Testing
Account,CN=Users,DC=pucit,DC=local'
```

Example 11

This example sets the notifications (in number of days) before group expiry. It also enables today's expiry reports as well as it enables the group attestation.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -EnableGroupAttestation -
GlmNotifyOwnersXDaysBeforeOperation add -
GlmNotifyOwnersXDaysBefore 1,3,10 -
GlmEnableNotificationOfTodaysExpiry
```

Example 12

This example creates a new role – *DemoRole1* – for the *AdStore9* identity store by specifying the minimum possible parameters.

By default, all permissions are declined to the role created through this commandlet. Moreover, no criteria filters or scope (group / container) are added to the role.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -RoleOperation add -RoleName
DemoRole1 -RolePriority 50 -RoleCriteriaScope Container
```

Example 13

This example creates a new security role – DemoRole1 – in *AdStore9* identity store and a container is set as its role criteria.

By default, all permissions are declined to the role created through this commandlet.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -RoleOperation add -RoleName
DemoRole1 -RolePriority 50 -RoleCriteriaScope Container -
RoleCriteriaDN 'ou=workingou,dc=pucit,dc=local'
```

Example 14

This example creates a new security role by specifying the container and criteria filters.

The value for RoleCriteriaFilters parameter is specified as 3-length arrays. At first index, specify the filter name which can be either 'name' or 'type'. Second index holds the operator which is one of the 'is exactly' and 'is not' operator. The third index of the array holds the client name or client type depending upon whether 'name' or 'type' is specified at the first index.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -RoleOperation add -RoleName
DemoRole4 -RolePriority 53 -RoleCriteriaScope Container -
RoleCriteriaDN 'ou=workingou,dc=pucit,dc=local' -
RoleCriteriaOperator Or -RoleCriteriaFilters @('name', 'is
exactly', 'automate arslanahmadvm'), @('type', 'is not',
'managementshell')
```

Example 15

This example creates a new security role by specifying the container, criteria filters and permissions. In this example, only Manage My Groups and Create User permissions are granted to the created role.

By default, all the permissions except those specified in RolePermissions parameter are denied to the role created through this commandlet.

The role permission names can be retrieved from Get-RolePermissionNames commandlet.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -RoleOperation add -RoleName
DemoRole6 -RolePriority 55 -RoleCriteriaScope Container -
RoleCriteriaDN 'ou=workingou,dc=pucit,dc=local' -
RoleCriteriaOperator Or -RoleCriteriaFilters @('name', 'is
exactly', 'automate arslanahmadvm'), @('type', 'is not',
'managementshell') -RolePermissions 'manage my groups',
'create user'
```

Example 16

This example creates a new security role by specifying a role criterion as container, criteria filters and permissions.

By default, all the permissions except those specified in RolePermissions parameter are granted. This is due to the presence of DefaultAllowPermissions. In this

example, only Manage my groups and Create user permissions are denied (and the remaining ones are granted) to the created role.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -RoleOperation add -RoleName
DemoRole5 -RolePriority 54 -RoleCriteriaScope Container -
RoleCriteriaDN 'ou=workingou,dc=pucit,dc=local' -
RoleCriteriaOperator Or -RoleCriteriaFilters @('name', 'is
exactly', 'automate arslanahmadvm'), @('type', 'is not',
'managementshell') -RolePermissions 'manage my groups',
'create user' –DefaultAllowRolePermissions
```

Example 17

This example creates two group prefixes – *dev* and *ment* –at identity store level.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -PrefixOperation add -Prefixes
'dev', 'ment'
```

Example 18

This example tracks history of the selected actions for the *AdStore9* identity store. The selected actions are additional owner change, expiration policy change and renewal of group. History retention period is also specified as last 120 days.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -HistoryTrackingOption
Selected_Actions -HistoryActionsOperation add -
HistorySelectedActions AdditionalOwnerChange,
ExpirationPolicyChange, GroupExpireRenew -HistoryRetention
Last_120_Days
```

Example 19

This example configures file and windows logging settings for the *AdStore9* identity store. File logging is set to Debug level and windows logging to FailureAudit.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -FileLoggingEvent Debug -
WindowsLoggingEvent FailureAudit
```

Example 20

This example configures out of bounds settings for the *AdStore9* identity store.

Maximum 500 members are allowed in each group and when the threshold reaches, the members will be nested into child groups. The orphan groups will be deleted. Do not update the membership and alert if the percentage in membership exceeds by more than 65% and either the current or new membership exceeds 200 members.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -MaximumMembersPerGroup 500 -
WhenGroupMembershipThresholdReach NestIntoChildGroups -
EnableOrphanGroupsDeletion -EnableOutOfBoundsAlerts -
MembershipCountThreshold 200 -MembershipPercentageThreshold
65
```

Example 21

This example configures profile validation settings. This example specifies that profile validation policies should be applied on group specified by distinguished name 'CN=ProfileValidation1,OU=ArslanAhmadOU,OU=WorkingOU,DC=pucit,DC=local'. Regular profiles should be validated within 60 days. New profiles should also be validated within 15 days. Validation extension period should be 7 days.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -ProfileValidationGroupDN
'CN=ProfileValidation1,OU=ArslanAhmadOU,OU=WorkingOU,DC=puc
it,DC=local' -RegularProfileValidationLifecycle 60 -
EnableNewProfileValidationLifecycle -
NewProfileValidationLifecycle 15 -
ProfileValidationExtensionPeriod 7
```

Example 22

This example configures profile validation settings. This example cmdlet adds two profile validation reminders. It also causes an attribute named 'info' to be updated with value 'Validation expired' when the profile validation is expired. It also causes the validation date to be removed after 5 days (after which the policies for new users are applied to the users).

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -
ProfileValidationReminderOperation add -
ProfileValidationReminders @('fourth', 45), @('fifth', 60)
-EnableAttributeUpdation -ProfileValidationAttributeName
info -ProfileValidationAttributeValue 'Validation expired'
-EnableValidationDateRemoval -ValidationDateRemovalInterval
5
```

Example 23

This example adds two security questions in the *AdStore9* identity store.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -QuestionOperation add -
SecurityQuestions 'When was the first time you felt that it
was raining even though it was not raining?', 'What would
happen if there were no GroupID?'
```

Example 24

This example configures password options.

This example enforces the following password policy: Do not allow passwords starting with either 'webdir123R' or containing '123R' and allow only those passwords matching '^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[^\da-zA-Z]).{8,15}$' regular expression pattern.

'PasswordExceptions' parameter accepts 2-Length arrays having at first index the operator and at second index the string. Allowed operators contain: 'equals'; 'startswith'; 'endswith'; 'contains'; and 'regexp'.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -PasswordExceptionOperation add
-PasswordExceptions @('startswith', 'webdir123R'),
@('contains', '123R') -PasswordRuleOperation add -
PasswordRules '^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[^\da-
zA-Z]).{8,15}$'
```

Example 25

This example configures the second way authentication via security questions.

```
Set-IdentityStore -IdentityStoreName AdStore9 -Credential
$creds -Domain pucit.local -
EnableSWAuthenticationViaSecurityQuestions -
SWAQuestionsOperation add -SWAQuestions @('when was the
first time you felt that it is raining even though it was
not raining?', 'info')
```

Set-IdentityStoreRole

Use the **Set-IdentityStoreRole** commandlet to modify properties of a security role in an identity store.

Syntax

```
Set-IdentityStoreRole
  -RoleName <string>
  -IdentityStoreName <string>
  [-NewName <string>]
  [-Description <string>]
  [-Priority <int>]
  [-Enabled <bool>]
  [-CriteriaScope {Group | Container}]
  [-DistinguishedName <string>]
  [-Operator {Or | And}]
  [-CriteriaFilters <string[][]>]
  [-FilterOperation {Add | Remove | RemoveAll}]
```



```
  [-Permissions <string[]>]
  [-PermissionOperation {GrantAll | GrantExcept | Grant |
Deny | DenyExcept | DenyAll}]
  [<CommonParameters>]
```

Required parameter

- RoleName
- IdentityStoreName

Example 1

This example modifies properties of the *DemoRole1* role in AdStore9 identity store. It renames the role to DemoRole1_Renamed and sets its priority to 45.

```
Set-IdentityStoreRole -RoleName DemoRole1 -
IdentityStoreName AdStore9 -NewName DemoRole1_Renamed -
Priority 45 -Enabled $True
```

Example 2

This example modifies the *DemoRole2* role in the AdStore9 identity store. The scope of the role is set to a container and removes filters specified in CriteriaFilters parameter.

```
Set-IdentityStoreRole -RoleName DemoRole2 -
IdentityStoreName AdStore9 -CriteriaScope Container -
DistinguishedName 'ou=workingou,dc=pucit,dc=local' -
FilterOperation Remove -CriteriaFilters @('type', 'is not',
'managementshell')
```

Example 3

This example modifies the permissions assigned to the DemoRole2 role. Two permissions *Manage any group* and *create smart group* are being assigned

```
Set-IdentityStoreRole -RoleName DemoRole2 -
IdentityStoreName AdStore9 -PermissionOperation Grant -
Permissions 'Manage any group', 'create smart group'
```

Set-MessagingServer

The commandlet **Set-MessagingServer** configures a messaging system in identity store. The SmtpServer parameter requires the server name of the messaging system to be specified. Get-AvailableMessagingServers commandlet can be used to retrieve the server names of the messaging systems.

This commandlet also has some parameters that appear depending on the value of the Provider parameter.

Syntax

```
Set-MessagingServer
  -IdentityStoreName <string>
  -Provider {o365 | gsuite | exchange2010 | exchange2013 |
exchange2016 | exchange2019}
  -Credential <pscredential>
  -SmtpServer <string>
  [-Priority <int>]
  [-Disabled]
  [<CommonParameters>]
```

```
Set-MessagingServer
  -IdentityStoreName <string>
  -Provider {o365 | gsuite | exchange2010 | exchange2013 |
exchange2016 | exchange2019}
  -UserName <string>
  -SmtpServer <string>
  [-Password <string>]
  [-Priority <int>]
  [-Disabled]
  [<CommonParameters>]
```

Required parameter

- IdentityStoreName
- Provider
- Credential
- SmtpServer

Example 1

This example modifies the messaging system of the *AsStore9* identity store to Microsoft Office 365.

```
Set-MessagingServer -IdentityStoreName AdStore9 -Provider
o365 -UserName admin@mydomain.onmicrosoft.com -Password
webdir123R -SmtpServer ps.outlook.com -Domain
mydomain.onmicrosoft.com -AppId 'a1b2c3d4-e5f6-f6e5-d4c3-
b2a1b2c3d4e5'
```

Example 2

This example modifies the messaging system of the *AdStore9* identity store to Google Apps (G-Suite).

```
Set-MessagingServer -IdentityStoreName AdStore9 -Provider
gsuite -UserName groupid@testproject-
219211.iam.gserviceaccount.com -AdminUsername
arslan@bibelotz.com -SmtpServer imap.gmail.com -
P12CertificatePath 'C:\Keys\gsuite\key.p12'
```

Example 3

This example modifies the messaging system of the *AdStore9* identity store to Exchange 2010.

```
Set-MessagingServer -IdentityStoreName AdStore9 -Provider
exchange2010 -UserName administrator -Password webdir123R -
SmtpServer arslanahmadsvm.pucit.local -Domain pucit.local
```

Set-Notifications

Use the **Set-Notifications** commandlet to modify the notification settings of an identity store.

Syntax

```
Set-Notifications
  -IdentityStoreName <string>
  [-PrimaryRecepients <string[]>]
  [-CarbonCopy <string[]>]
  [-NotifyLoggedInUsers <bool>]
  [-NotifyOwners <bool>]
  [-NotifyModifiedObject <bool>]
  [-NotifyPublicGroupOwner <bool>]
  [-NotifyAddedMembers <bool>]
  [-PasswordPortalUrl <string>]
  [-NotifyUserGroupJoinML <bool>]
  [-NotifyUserGroupLeaveML <bool>]
  [-XDaysBeforeLeaveNotificationML <int>]
  [-NotifyUserGroupJoinMB <bool>]
  [-NotifyUserGroupLeaveMB <bool>]
  [-XDaysBeforeLeaveNotificationMB <int>]
  [<CommonParameters>]
```

Required parameter

- IdentitySoreName

Example 1

This example sets the primary and carbon copy (CC) recipients of the notifications for the *AdStore9* identity store. Additionally, it also sets the group owners / managers and public group owners as the notification recipients.

```
Set-Notifications -IdentityStoreName AdStore9 -
PrimaryRecepients 'euser1@pucit.local',
'euser2@pucit.local' -CarbonCopy 'exmb1@pucit.local' -
NotifyOwners $true -NotifyPublicGroupOwner $true
```

Example 2

This example configures recipients for membership lifecycle notifications i.e. it notifies the user upon joining a group and intimates the user before 7 days it is removed as a member from the group.

```
Set-Notifications -IdentityStoreName AdStore9 -
NotifyUserGroupJoinML $true -XDaysBeforeLeaveNotificationML
7
```

Set-SmtpServer

The **Set-SmtpServer** commandlet configures an SMTP server for an identity store.

Syntax

```
Set-SmtpServer
  -IdentityStorename <string>
  -SmtpServer <string>
  -FromEmail <string>
  -ToEmail <string>
  -Port <int>
  [-Credential <pscredential>]
  [-UseSmptUserAuthentication]
  [-SslEnabled]
  <CommonParameters>]
```

Required parameter

- IdentityStorename
- SmtpServer
- FromEmail
- ToEmail
- Port

Example 1

This example configures *arsalanahmadsvm.pucit.local* SMTP server for *AdStore9* identity store on port 25. Email address for sending notification is specifies as noreply@pucit.local and euser1@pucit.local as recipient email address.

```
Set-SmtpServer -IdentityStorename AdStore9 -SmtpServer
arslanahmadsvm.pucit.local -FromEmail noreply@pucit.local -
ToEmail euser1@pucit.local -Port 25
```

Example 2

This example configures *arsalanahmadsvm.pucit.local* SMTP server that is SSL (Secured Socket Layer) enabled for *AdStore* identity store. The SMTP server is configured on port 555 and it uses credentials stored in the $creds variable.

```
Set-SmtpServer -IdentityStorename AdStore9 -SmtpServer
smtp.office365.com -FromEmail
admin@mydomain.onmicrosoft.com -ToEmail
admin@mydomain.onmicrosoft.com -Port 555 -
UseSmptUserAuthentication -SslEnabled -Credential $creds
```

Chapter 5 - User Commands

This chapter covers commandlets for performing user related tasks such as:

- Get-User: retrieves user that match the given criteria.
- Get-UserEnrollment: displays information about the status of user enrollment.
- New-User: creates a new user.
- Remove-User: removes a user from Directory.
- Set-User: modifies a user in Directory

Review the description of the supported parameters of these commandlets along with their attributes in the List of Parameters table.

Common parameters of Windows Management Shell are not supported in GroupID Management Shell.

Get-User

Use the **Get-User** commandlet to retrieve basic information about a user that match your given criteria.

Syntax

```
Get-User
  [[-Identity] <string[]>]
  [-SearchContainer <string[]>]
  [-SearchContainersScopeList <string>]
  [-ShouldReturnCollection]
  [-MaxItemsToDisplay <int>]
  [-ObjectType <string[]>]
  [-LdapFilter <string>]
  [-SmartFilter <string>]
  [-ServerFilter <string>]
  [-AttributesToLoad <string[]>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- None

Example

The following command retrieves the specified user from the connected identity store.

```
Get-User -Identity "Osama"
```

Get-UserEnrollment

The commandlet **Get-UserEnrollment** retrieves enrollment information of a user.

Syntax

```
Get-UserEnrollment
  -Identity <string>
  [-EnrollmentTypes {None | Mobile | SecurityQuestions |
Email | Authenticator | LinkAccount | PhoneID | Yubikey |
WindowsHello | All | Any}]
  [<CommonParameters>]
```

Required parameter

- Identity

Example 1

If user is enrolled, this cmdlet will enlist the authentication type(s) the user is enrolled with.

```
Get-UserEnrollment -Identity euser1
```

Example 2

Check whether the specified user is enrolled in the specified enrollment type(s).

```
Get-UserEnrollment -Identity euser1 -EnrollmentTypes
SecurityQuestions, Email
```

Example 3

This example gets user enrollment information through the pipeline operator.

```
'euser1', 'euser2' | Get-UserEnrollment
```

New-User

Use the **New-User** commandlet to create a new user in Directory. Most user properties can be directly added by using the parameters of this commandlet.

Syntax

```
New-User
  -Name <string>
  -OrganizationalUnit <string>
  -SAMAccountName <string>
  -Password <string>
  -FirstName <string>
  -LastName <string>
  -DisplayName <string>
  [-UPNSuffix <string>]
  [-Title <string>]
  [-City <string>]
  [-State <string>]
  [-Zip <string>]
  [-Country <string>]
  [-Initials <string>]
  [-Address <string>]
  [-Office <string>]
  [-Business <string>]
  [-Business2 <string>]
  [-Alias <string>]
  [-EmailAddress <string>]
  [-Department <string>]
  [-Company <string>]
  [-Mobile <string>]
  [-Home <string>]
  [-AccountDisabled <string>]
  [-PasswordNeverExpires <string>]
  [-PasswordForceChange <string>]
  [-Manager <string[]>]
  [-HomePage <string>]
  [-Assistant <string>]
  [-Notes <string>]
  [-MailEnabled <string>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameters

- Name
- OrganizationalUnit

- SAMAccountName
- Password
- FirstName
- LastName
- DisplayName

Example

The following command creates a new user in the container specified by the **OrganizationalUnit** parameter. The command also specifies the logon name, password, first name, last name and display name of the new user.

```
New-User -Name "OsamaUser" -OrganizationalUnit
"OU=osamamu,DC=naveed,DC=local" -SAMAccountName
"OsamaUser11" -Password "webdir123R" -FirstName "Osama" -
LastName "Shahbaz" -DisplayName "Osama"
```

Remove-User

Use the **Remove-User** commandlet to delete user from directory.

Syntax

```
Remove-User
  -Identity <string[]>
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example

The following command deletes a user with the specified name.

```
Remove-User -Identity "osama"
```

Set-User

The **Set-User** commandlet modifies a user in Directory. Most user properties can be directly modified by using the parameters of this commandlet.

Syntax

```
Set-User
  -Identity <string>
  [-FirstName <string>]
  [-LastName <string>]
  [-Title <string>]
  [-City <string>]
  [-State <string>]
  [-Zip <string>]
  [-Country <string>]
  [-Initials <string>]
  [-Address <string>]
  [-Office <string>]
  [-Business <string>]
  [-Add <hashtable[]>]
  [-Remove <hashtable[]>]
  [-Replace <hashtable[]>]
  [-Clear <string[]>]
  [-Department <string>]
  [-Company <string>]
  [-Assistant <string>]
  [-HomePage <string>]
  [-Alias <string>]
  [-EmailAddress <string>]
  [-Description <string>]
  [-Notes <string>]
  [-AdministrativeNotes <string>]
  [-DisplayName <string>]
  [-SimpleDisplayName <string>]
  [-CustomAttribute1 <string>]
  [-CustomAttribute2 <string>]
  [-CustomAttribute3 <string>]
  [-CustomAttribute4 <string>]
  [-CustomAttribute5 <string>]
  [-CustomAttribute6 <string>]
  [-CustomAttribute7 <string>]
  [-CustomAttribute8 <string>]
  [-CustomAttribute9 <string>]
  [-CustomAttribute10 <string>]
  [-CustomAttribute11 <string>]
  [-CustomAttribute12 <string>]
  [-CustomAttribute13 <string>]
  [-CustomAttribute14 <string>]
  [-CustomAttribute15 <string>]
```



```
  [-Delimiter <string>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example

The following command modifies the display name of the specified user.

```
Set-User -Identity "Osama" -DisplayName "Osama123"
```

Chapter 6 - User Lifecycle Commands

User lifecycle process ensures the accuracy of users' information in the directory, this chapter covers commandlets for performing user lifecycle related tasks such as:

- Extend-User: extend the user lifecycle for specified period of days of an expired user.
- Get-Status: provides the status of specified user as per the profile validation criteria.
- Reinstate-User: activates or disables a user.
- Terminate-DirectReports: terminates direct reports of user.
- Transfer-DirectReports: transfers direct reports of user.

Review the description of the supported parameters of these commandlets along with their attributes in the List of Parameters table.

Common parameters of Windows Management Shell are not supported in GroupID Management Shell.

Extend-User

Use the **Extend-User** commandlet to extend the user lifecycle of an expired user for specified period of days for the connected identity store.

Syntax

```
Extend-User
  -Identity <string[]>
  [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <PSCredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example

The following commandlet extends the profile validation period for the specified user. The period is extended up to the specified days for the connected identity store.

```
Extend-User -Identity
"CN=ImanamiUser100,OU=BulkUsers,DC=gid,DC=local"
```

Get-Status

Use the **Get-Status** command to know the status of specified user as per the profile validation criteria defined for the connected identity store.

Syntax

```
Get-Status
  -Manager <string>
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Manager

Example

The following command provides information about the status of the specified user as per the criteria defined for user life cycle for the connected identity store. It also provides information on the number of days left to validate the profile again.

```
Get-Status –Manager "Richard"
```

Reinstate-User

Use the **Reinstate-User** command to activate or disable a user. Users can be disabled for any of the following reasons:

- Users that have been disabled for not validating their profiles within the required period.
- Users that have been terminated or disabled by their respective managers.
- Users that are disabled in the directory.

An administrator or member of Helpdesk role can reinstate a disabled user.

Syntax

```
Reinstate-User
  -Identity <string>
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example

The following command reinstates the specified disabled of the connected identity store.

```
Reinstate-User -Identity "Farzana Jafar"
```

Terminate-DirectReports

Use the **Terminate-DirectRreports** command to terminate user(s). Specify manager of the user you want to terminate.

You can perform this function in directory as per your role and permissions.

Syntax

```
Terminate-DirectReports
  -DirectReports <string[]>
  -Manager <string>
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameters

- DirectReports
- Manager

Example

The following command terminates the specified users in the connected identity store. Their Manager is also specified in the command who will receive notification as per the defined workflow.

```
Terminate-DirectReports -DirectReports "Irfan","Naeem" -
Manager "Raja"
```

Transfer-DirectReports

Use the **Transfer-DirectReports** commandlet to transfer direct report(s) in the connected identity store. Specify manager who will approve this transfer.

You can perform this function in directory as per your role and permissions.

Syntax

```
Transfer-DirectReports
  -DirectReports <string[]>
  -Manager <string>
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameters

- DirectReports
- Manager

Example

The following command transfers two direct reports to Manager Robin.

```
Transfer-DirectReports -DirectReports "F Jafar","azram" -
Manager "Robin"
```

Chapter 7 - Contact Commands

This chapter covers commandlets for performing contact related tasks such as:

- Get- Contact: retrieves a contact that match the given criteria.
- New-Contact: creates a new contact.
- Remove- Contact: removes a contact from Directory.
- Set- Contact: modifies a contact in Directory.

Review the description of the supported parameters of these commandlets along with their attributes in the List of Parameters table.

Common parameters of Windows Management Shell are not supported in GroupID Management Shell.

Get-Contact

Use the **Get-Contact** commandlet to retrieve basic information about a contact that match your given criteria.

Syntax

```
Get-Contact
  [[-Identity] <string[]>]
  [-SearchContainer <string[]>]
  [-SearchContainersScopeList <string>]
  [-ShouldReturnCollection]
  [-MaxItemsToDisplay <int>]
  [-ObjectType <string[]>]
  [-LdapFilter <string>]
  [-SmartFilter <string>]
  [-ServerFilter <string>]
  [-AttributesToLoad <string[]>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- None

Example

The following command retrieves contact from the specified container of the connected identity store.

```
Get-Contact -SearchContainer
"OU=osamamu,DC=naveed,DC=local"
```

New-Contact

Use the **New-Contact** commandlet to create a new contact in Directory. Most contact properties can be directly added by using the parameters of this commandlet.

Syntax

```
New-Contact
  -Name <string>
  -OrganizationalUnit <string>
  -FirstName <string>
  -LastName <string>
  -DisplayName <string>
  [-UPNSuffix <string>]
  [-Title <string>]
  [-City <string>]
  [-State <string>]
  [-Zip <string>]
  [-Country <string>]
  [-Initials <string>]
  [-Address <string>]
  [-Office <string>]
  [-Business <string>]
  [-Business2 <string>]
  [-Alias <string>]
  [-EmailAddress <string>]
  [-Department <string>]
  [-Company <string>]
  [-Mobile <string>]
  [-Home <string>]
  [-Manager <string[]>]
  [-HomePage <string>]
  [-Assistant <string>]
  [-Notes <string>]
  [-MailEnabled <string>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameters

- Name
- OrganizationalUnit
- FirstName
- LastName
- DisplayName

Example

The following command creates a new contact in the container specified by the **OrganizationalUnit** parameter. The command also specifies the logon name, first name, last name and display name of the new contact.

```
New-Contact -Name "OsamaContact" -OrganizationalUnit
"OU=osamamu,DC=naveed,DC=local" -FirstName "OsamaContact" -
LastName "OsamaContact" -DisplayName "OsamaContact"
```

Remove-Contact

Use the **Remove-Contact** commandlet to delete contact from Directory.

Syntax

```
Remove-Contact
  -Identity <string[]>
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example

The following command deletes the specified contact from the connected identity store.

```
Remove-Contact -Identity "OsamaContact"
```

Set-Contact

The **Set-User** commandlet modifies a user in Directory. Most user properties can be directly modified by using the parameters of this commandlet.

Syntax

```
Set-Contact
  -Identity <string>
  [-FirstName <string>]
  [-LastName <string>]
  [-Title <string>]
  [-City <string>]
  [-State <string>]
  [-Zip <string>]
  [-Country <string>]
  [-Initials <string>]
  [-Address <string>]
  [-Office <string>]
  [-Business <string>]
  [-Add <hashtable[]>]
  [-Remove <hashtable[]>]
  [-Replace <hashtable[]>]
  [-Clear <string[]>]
  [-Department <string>]
  [-Company <string>]
  [-Assistant <string>]
  [-HomePage <string>]
  [-Alias <string>]
  [-EmailAddress <string>]
  [-Description <string>]
  [-Notes <string>]
  [-AdministrativeNotes <string>]
  [-DisplayName <string>]
  [-SimpleDisplayName <string>]
  [-CustomAttribute1 <string>]
  [-CustomAttribute2 <string>]
  [-CustomAttribute3 <string>]
  [-CustomAttribute4 <string>]
  [-CustomAttribute5 <string>]
  [-CustomAttribute6 <string>]
  [-CustomAttribute7 <string>]
  [-CustomAttribute8 <string>]
  [-CustomAttribute9 <string>]
  [-CustomAttribute10 <string>]
  [-CustomAttribute11 <string>]
  [-CustomAttribute12 <string>]
  [-CustomAttribute13 <string>]
  [-CustomAttribute14 <string>]
  [-CustomAttribute15 <string>]
```

```
  [-Delimiter <string>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example

The following command modifies the city of the specified contact.

```
Set-Contact -Identity "OsamaContact" -City "Islamabad"
```

Chapter 8 - Group Commands

This chapter covers commandlets for performing tasks related to managed and unmanaged groups:

- Convert-Group: converts an unmanaged group to a Smart Group.
- ConvertTo-StaticGroup:
- Expire Group: expires the group temporarily.
- Get-Group: retrieves groups from one or more containers.
- Move-Group: moves a group to a different container in the same domain or in a different domain.
- New-Group: creates an unmanaged group.
- Remove-Group: deletes a managed or unmanaged group or Dynasty in Directory.
- Renew Group: reactivates an expired group.
- Set-Group: modifies an unmanaged group in Directory.

Review the description of the supported parameters of these commandlets along with their attributes in the List of Parameters**Error! Reference source not found.** table.

Common parameters of Windows Management Shell are not supported in GroupID Management Shell.

Convert-Group

This command converts an unmanaged group to a Smart Group.

GroupID Management Shell prompts for the identity of the unmanaged group you want to convert into a Smart Group. After executing the commandlet displays the status that update is successful as shown in the following snapshot.

```
[PS GID] C:\WINDOWS\system32>convert-group -Identity '299e9c59-2ac9-4974-a3ba-26869c764303'

Status   :: Upgrade is Successful



DistinguishedName  : CN=asd12,OU=\<ou\>,DC=mirza,DC=local
DisplayName        : asd12
Description        : a
ManagedBy_Resolved : CN=Administrator,CN=Users,DC=mirza,DC=local
ObjectClass        : group
ObjectGuid         : 299e9c59-2ac9-4974-a3ba-26869c764303
SAMAccountName     : asd12
WhenCreated        : 2019 July 05 08:22:26
UserPrincipalName  :
Name               : asd12
AdditionalOwner    :
Criteria           :
Member             : {Administrator}
MembershipType     :
ManagedGroupType   : 2
SAMAccountType     :
WhenChanged        : 2019 July 25 06:46:03
ModifyTimestamp    : 2019 July 25 06:46:03
Cn                 : asd12
ExpirationDate     : 9999 December 31 00:00:00
ExpirationPolicy   : 0
IsExpired          : False
Security           : Private
Sid                : S-1-5-21-3653390206-1211385926-1943145492-1163
ObjectCategory     : CN=Group,CN=Schema,CN=Configuration,DC=mirza,DC=local
EmailAddress       :
AdsPath            :
```

Figure 4: GroupID Management Shell window

The converted Smart Group will not have an LDAP query attached to it. You have to define it manually.

Syntax

```
Convert-Group
  -Identity <string>
  [-SearchContainers <string[]>]
  [-SearchContainersScopeList <string[]>]
  [-ObjectTypes <string[]>]
  [-LdapFilter <string>]
  [-IncludeRecipients <string[]>]
  [-ExcludeRecipients <string[]>]
  [-Storage <string>]
  [-DataSourceType <string>]
  [-SystemDSN <string>]
  [-TableOrView <string>]
  [-DataSourceUserName <string>]
  [-DataSourcePassword <string>]
  [-FilePath <string>]
  [-Server <string>]
  [-Port <int>]
```

```
[-LDAPSearchContainer <string>]
[-DataSourceName <string>]
[-DataSourceQuery <string>]
[-WindowsAuthentication]
[-EnableUpdate <string>]
[-IsPasswordExpirySmartDL]
[-ExpirationRange <int>]
[-DomainExpiration <int>]
[-MaximumPasswordAge <int>]
[-MinimumPasswordAge <int>]
[-IncludeDisabledUsers <string>]
[-IncludePasswordNeverExpireUsers <string>]
[-SendEmail <string>]
[-EmailTemplatePath <string>] [-Script <string>]
[-ScriptFilePath <string>]
[-Provider_Container <string>]
[-PowerTools <ArrayList>]
[-KeyMapAD <string>]
[-KeyMapDB <string>]
[-ExtendGroupLife]
[-ExpirationPolicy <int>]
[-MsExchCoManagedByLink <string[]>]
[-IsExpired <string>]
[-GroupScope <string>]
[-Type <string>]
[-Prefix <string>]
[-SecurityType <string>]
[-ManagedBy <string[]>]
[-MaxSendSize <int>]
[-AcceptMessagesOnlyFrom <string[]>]
[-RejectMessagesFrom <string[]>]
[-AcceptMessagesOnlyFromGroups <string[]>]
[-RejectMessagesFromGroup <string[]>]
[-AdditionalOwners <string[]>]
[-NotifyOptOutAdditionalOwners <string[]>]
[-ExpansionServer <string>]
[-BypassOwnersPolicy <string>]
[-MsExchRequireAuthToSendTo <string>]
[-HiddenFromAddressListEnabled <string>]
[-SendOofMessageToOriginatorEnabled <string>]
[-HideMembershipFromAddressListEnabled <string>]
[-ReportToManagerEnabled <string>]
[-ReportToOriginatorEnabled <string>]
[-UpdateMembershipByManagerEnabled <string>]
[-Add <hashtable[]>]
[-Remove <hashtable[]>]
[-Replace <hashtable[]>]
[-Clear <string[]>]
[-Department <string>]
[-Company <string>]
[-Assistant <string>]
[-HomePage <string>]
```

```
  [-Alias <string>]
  [-EmailAddress <string>]
  [-Description <string>]
  [-Notes <string>]
  [-AdministrativeNotes <string>]
  [-DisplayName <string>]
  [-SimpleDisplayName <string>]
  [-CustomAttribute1 <string>]
  [-CustomAttribute2 <string>]
  [-CustomAttribute3 <string>]
  [-CustomAttribute4 <string>]
  [-CustomAttribute5 <string>]
  [-CustomAttribute6 <string>]
  [-CustomAttribute7 <string>]
  [-CustomAttribute8 <string>]
  [-CustomAttribute9 <string>]
  [-CustomAttribute10 <string>]
  [-CustomAttribute11 <string>]
  [-CustomAttribute12 <string>]
  [-CustomAttribute13 <string>]
  [-CustomAttribute14 <string>]
  [-CustomAttribute15 <string>]
  [-Delimiter <string>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example

The following commandlet converts a group Clay2 group to a Smart Group using the credentials of current logged-on user.

```
Convert-Group -Identity "Clay2" -Credential $Cred
```

Expire-Group

The **Expire-Group** commandlet expires a group temporarily. All notifications to the expired group will be stopped and all group related functionalities will be on halt.

You can view events related to this commandlet in GroupID Management Console, on the History tab of the object's Properties dialog box.

Syntax

```
Expire-Group
  -Identity <string[]>
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example

The following command expires the specified group from the connected identity store.

```
Expire-Group –Identity "CN=Training,OU=Local
Recruiting,DC=HR,DC=Imanami,DC=US"
```

Get-Group

This **Get-Group** commandlet retrieves both managed and unmanaged groups that are in one or more containers on a domain matching the given criteria.

Syntax

```
Get-Group
  [[-Identity] <string[]>]
  [-SearchContainer <string[]>]
  [-SearchContainersScopeList <string>]
  [-ShouldReturnCollection]
  [-MaxItemsToDisplay <int>]
  [-ObjectType <string[]>]
  [-LdapFilter <string>]
  [-SmartFilter <string>]
  [-ServerFilter <string>]
  [-AttributesToLoad <string[]>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- None

Example 1

The following command retrieves all groups in the base container specified by the **SearchContainer** parameter including sub-containers, using the credentials of current user logged-on to the identity store.

```
Get-Group -SearchContainer
"OU=Recuriting,DC=HR,DC=Imanami,DC=US"
```

Example 2

The following command retrieves all groups with a display name beginning with the **S** in the base containers specified by the **SearchContainer** parameter including sub-containers of the first base container and excluding sub-containers of the second one using the credentials set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A.

```
Get-Group -SearchContainer
"OU=Recuriting,DC=HR,DC=Imanami,DC=US","OU=OutSourcing,DC=H
R,DC=Imanami,DC=US" -SearchContainersScopeList "2","1" -
LdapFilter "(DisplayName = S*)" -Credential $Cred
```

Example 3

The following command retrieves all Smart Groups from the connected identity store with Security Type **Private** and **John Smith** as their additional owner. The **OUT-NULL** commandlet is useful for preventing the retrieved groups' information from appearing on the console.

```
Get-Group -SmartFilter "(SecurityType = Private)" | Set-
Group -AdditionalOwners
"CN=JohnSmith,DC=HR,DC=Imanami,DC=US" | OUT-NULL
```

Move-Group

The **Move-Group** commandlet enables you to move a group to a different container in the same domain or in a different domain within the same forest. Movement of groups across forests is not allowed.

You can view events related to this commandlet in GroupID Management Console, on the History tab of the object's Properties dialog box.

Syntax

```
Move-Group
  -Identity <string>
  -DestinationContainer <string>
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameters

- Identity
- DestinationContainer

Example 1

The following command moves the group **Training** to the **Local Recruiting** organizational unit using the credentials of current user logged-on to the identity store.

```
Move-Group -Identity
"CN=Training,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -
DestinationContainer "OU=Local
Recruiting,OU=Recruiting,DC=HR,DC=Imanami,DC=US"
```

Example 2

The following command moves the group **Training** to the **OffShore Recruiting** organizational unit. The command uses the credentials set in the **$Credentials** environment variable for moving a group. For information about setting credentials, see Appendix A.

```
Move-Group -Identity "CN=Training,OU=Local
Recruiting,DC=HR,DC=Imanami,DC=US" -DestinationContainer
"OU=OffShore
Recruiting,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -
Credential $Cred
```

New-Group

Use the **New-Group** commandlet to create a new unmanaged group in a particular container in directory.

You can view events related to this commandlet in GroupID Management Console, on the History tab of the object's Properties dialog box.

Syntax

```
New-Group
  -SamAccountName <string>
  -Name <string>
  -OrganizationalUnit <string>
  -GroupScope <string>
  -Type <string>
  -SecurityType <string>
  [-GroupAlias <string>]
  [-ManagedBy <string[]>]
  [-DisplayName <string>]
  [-MailEnabled <string>]
  [-Description <string>]
  [-AdditionalOwners <string[]>]
  [-NotifyOptOutAdditionalOwners <string[]>]
  [-Members <string[]>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameters

- SamAccountName
- Name
- OrganizationalUnit
- GroupScope
- Type
- SecurityType

Example 1

The following command creates a new unmanaged, mail-disabled, global, distribution group in the container specified by the **OrganizationalUnit** parameter, using the credentials of current user logged-on to the identity store.

```
New-Group -Name "Event Management" -OrganizationalUnit
"OU=Local Recruiting,OU=Recruiting,DC=HR,DC=Imanami,DC=US"
-GroupAlias "EventManagement" -SamAccountName "Event
Management" -GroupScope "Global Group" -Type "Distribution"
```

Example 2

The command below creates a new mail-enabled, domain-local, semi-private, security group in the container specified by the **OrganizationalUnit** parameter, using the credentials set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A.

```
New-Group -Name "Enrollment" -OrganizationalUnit "OU=Local
Recruiting,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -
GroupAlias "Enrollment" -MailEnable True -SamAccountName
"Enrollment" -GroupScope "Domain Local" -Type "Security" -
SecurityType "Semi_Private"
```

Remove-Group

Use this commandlet to delete a group (managed or unmanaged) or Dynasty in directory. Removing a parent Dynasty using this commandlet removes all its children as well.

You can view events related to this commandlet in GroupID Management Console, on the History tab of the object's Properties dialog box.

Syntax

```
Remove-Group
  -Identity <string[]>
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example 1

The following command removes the **Event Management** group, using the credentials of current user logged-on to the identity store.

```
Remove-Group -identity "OU=Event
Management,OU=Recruiting,DC=HR,DC=Imanami,DC=US"
```

Example 2

The following command first shows the changes that will be made by executing the command (a deletion). The command uses the credentials set in the **$Credentials** environment variable to perform the deletion. For information about setting credentials, see Appendix A.

```
Remove-Group -identity "OU=Event
Management,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -
Credential $Cred
```

Renew-Group

The **Renew-Group** commandlet re-activates an expired group.

You can view events related to this commandlet in GroupID Management Console, on the History tab of the object's Properties dialog box.

Syntax

```
Renew-Group
  -Identity <string[]>
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example

The following command renews the specified group in the connected identity store.

```
Renew-Group -Identity "CN=Training,OU=Local
Recruiting,DC=HR,DC=Imanami,DC=US"
```

Set-Group

The **Set-Group** commandlet modifies an unmanaged group in directory. However, you can use this commandlet to modify those parameters of a Smart Group that are native attributes of an unmanaged group in Directory. For modifying Smart Group-specific attributes, you can use the Set-SmartGroup commandlet.

You can view events related to this commandlet in GroupID Management Console, on the History tab of the object's Properties dialog box.

Syntax

```
Set-Group
  -Identity <string>
  [-ExtendGroupLife]
  [-ExpirationPolicy <int>]
  [-MsExchCoManagedByLink <string[]>]
  [-IsExpired <string>]
  [-GroupScope <string>]
  [-Type <string>]
  [-Prefix <string>]
  [-SecurityType <string>]
  [-ManagedBy <string[]>]
  [-MaxSendSize <int>]
  [-AcceptMessagesOnlyFrom <string[]>]
  [-RejectMessagesFrom <string[]>]
  [-AcceptMessagesOnlyFromGroups <string[]>]
  [-RejectMessagesFromGroup <string[]>]
  [-AdditionalOwners <string[]>]
  [-NotifyOptOutAdditionalOwners <string[]>]
  [-ExpansionServer <string>]
  [-BypassOwnersPolicy <string>]
  [-MsExchRequireAuthToSendTo <string>]
  [-HiddenFromAddressListEnabled <string>]
  [-SendOofMessageToOriginatorEnabled <string>]
  [-HideMembershipFromAddressListEnabled <string>]
  [-ReportToManagerEnabled <string>]
  [-ReportToOriginatorEnabled <string>]
  [-UpdateMembershipByManagerEnabled <string>]
  [-Add <hashtable[]>]
  [-Remove <hashtable[]>]
  [-Replace <hashtable[]>]
  [-Clear <string[]>]
  [-Department <string>]
  [-Company <string>]
  [-Assistant <string>]
  [-HomePage <string>]
  [-Alias <string>]
  [-EmailAddress <string>]
  [-Description <string>]
  [-Notes <string>]
  [-AdministrativeNotes <string>]
  [-DisplayName <string>]
  [-SimpleDisplayName <string>]
  [-CustomAttribute1 <string>]
  [-CustomAttribute2 <string>]
  [-CustomAttribute3 <string>]
  [-CustomAttribute4 <string>]
  [-CustomAttribute5 <string>]
  [-CustomAttribute6 <string>]
  [-CustomAttribute7 <string>]
```

```
  [-CustomAttribute8 <string>]
  [-CustomAttribute9 <string>]
  [-CustomAttribute10 <string>]
  [-CustomAttribute11 <string>]
  [-CustomAttribute12 <string>]
  [-CustomAttribute13 <string>]
  [-CustomAttribute14 <string>]
  [-CustomAttribute15 <string>]
  [-Delimiter <string>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example 1

The following command changes the expiration policy of the **Training** group to 60 days and assigns a manager to the group, using the credentials of current user logged-on to the identity store.

```
Set-Group -Identity
"CN=Training,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -
ExpirationPolicy "60" -ExtendGroupLife -ManagedBy "CN=John
Smith,CN=Users,DC=HR,DC=Imanami,DC=US"
```

Example 2

The following command expires the group **Training**, using the credentials set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A.

```
Set-Group -Identity
"CN=Training,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -
IsExpired True -Credential $Cred
```

Example 3

The following command gets all groups in the container **Recruiting**, clears their additional owner lists and sets their expiration policy to **Never Expire**. The **OUT-NULL** commandlet has been used to prevent the retrieved groups information from appearing on the console.

```
Get-Group -searchcontainer
"OU=Recruiting,DC=HR,DC=Imanami,DC=US" | Set-Group -
AdditionalOwners "" -ExpirationPolicy "0" -ExtendGroupLife
| OUT-NULL
```

Example 4

The following command removes two additional owners from the **Training** group and adds three new additional owners to the group and excludes an additional owner from receiving e-mail notifications.

```
Set-Group -Identity
"CN=Training,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -Remove
@{AdditionalOwners =
"CN=Roger_Manson,OU=ResignedStaff,DC=HR,DC=Imanami,DC=US","
KillenEdward"} -Add @{AdditionalOwners =
"RobinSoto","MeganFox","DollyChan"} -
NotifyOptOutAdditionalOwners "RobinSoto"
```

Chapter 9 - Smart Group Commands

This chapter covers commandlets for managing Smart Groups and GroupID configuration settings. The commandlets are:

- ConvertTo-StaticGroup: converts a Smart Group or a dynasty into a static group.
- Get-Options: retrieves GroupID configuration settings and their corresponding values.
- Get-SmartGroup: retrieves Smart Groups and Dynasties that match the given criteria.
- New-SmartGroup: creates a new Smart Group (managed group) in Directory.
- Set-Options: modifies the values of GroupID configuration settings.
- Set-SmartGroup: modifies a Smart Group in Directory.
- Update-Group: modifies the membership of a Smart Group or Dynasty according to the results returned by the LDAP query.
- Upgrade-Group: upgrades managed (Smart Groups and Dynasties) and non-managed Groups of GroupID versions 7, 8 and 9 to GroupID 10.0.

Review the description of the supported parameters of these commandlets along with their attributes in the List of Parameters**Error! Reference source not found.** table.

Common parameters of Windows Management Shell are not supported in GroupID Management Shell.

ConvertTo-StaticGroup

The **ConvertTo-StaticGroup** commandlet converts an existing Smart Group or a dynasty to a static group by removing the attributes of the Smart Group or the dynasty.

Syntax

```
ConvertTo-StaticGroup
  -IdentityStoreName <string>
  [-GroupName <string[]>]
  [-SearchContainers <string[]>]
  [<CommonParameters>]
```

Required parameter

- IdentityStoreName

Example 1

The following commandlets converts a Smart Group in AdStore9 identity store *Smart_Training* to a static group.

```
ConvertTo-StaticGroup -IdentityStoreName AdStore9 -
GroupName "Smart_Training" -SearchContainers
"OU=Recruiting,OU=HR,DC=Imanami,DC=US","OU=Outsourcing,OU=H
R,DC=Imanami,DC=US"
```

Get-Options

GroupID stores its configuration settings in your system registry. The **Get-Options** commandlet helps you to retrieve these settings along with their corresponding values.

Syntax

```
Get-Options
  [[-Options] <string[]>]
  [-AttributesToLoad <string[]>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- None

Example 1

The following cmdlet lists down the specified GroupID settings with their values, from the registry, using the credentials of current user logged-on to the identity store.

```
Get-Options –Options "DLPrefix", "SMTPServer",
"NotificationFrom"
```

Example 2

The following command lists the options from the registry specified by the **Options** parameter, using the credentials set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A.

```
Get-Options –Options "DLPrefix", "SMTPServer",
"NotificationFrom" –Credential $Cred
```

Get-SmartGroup

Use this commandlet to retrieve Smart Groups and Dynasties that match your given criteria in one or more containers on a domain.

Syntax

```
Get-SmartGroup
  [[-Identity] <string[]>]
  [-SmartGroupType <string>]
  [-TopLevelOnly <bool>]
  [-GroupIDVersion <string>]
  [-SearchContainer <string[]>]
  [-SearchContainersScopeList <string>]
  [-ShouldReturnCollection]
  [-MaxItemsToDisplay <int>]
  [-ObjectType <string[]>]
  [-LdapFilter <string>]
  [-SmartFilter <string>]
  [-ServerFilter <string>]
  [-AttributesToLoad <string[]>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```


Required parameter

- None

Example 1

The following command retrieves only Smart Groups (not Dynasties) in the base container specified by the **SearchContainer** parameter including sub-containers, using the credentials of current user logged-on to the identity store.

```
Get-SmartGroup -SmartGroupType "SmartGroup" -
SearchContainer "OU=Recuriting,DC=HR,DC=Imanami,DC=US"
```

Example 2

The following command retrieves both Smart Groups and Dynasties that have display names starting with **S** in the containers specified by the **SearchContainer** parameter including sub-containers of the first base container and excluding sub-containers of the second one, using the credentials specified in the **$Credentials** environment variable. For information about setting credentials, see Appendix A.

```
Get-SmartGroup -SearchContainer
"OU=Recuriting,DC=HR,DC=Imanami,DC=US","OU=OutSourcing,DC=H
R,DC=Imanami,DC=US" -SearchContainersScopeList "2","1" -
LdapFilter "(DisplayName = S*)" -Credential $Cred
```

New-SmartGroup

This commandlet helps you to create a new Smart Group (managed group) in Directory. A Smart Group is a conventional distribution or security group that dynamically maintains its membership based on the rules applied by a user-defined LDAP query.

A Smart Group can also be defined as a Password Expiry group. A Password Expiry group is a dynamic group whose membership is based on password policy conditions defined by the administrator. The LDAP query defined for a Smart Group can be updated any time using the Set-SmartGroup commandlet. When the LDAP query is changed, you must update the group once to modify its membership according to the changes made to the query. For information about updating a group, see Update-Group.

You can view events related to this commandlet in GroupID Management Console, on the History tab of the object's Properties dialog box.

Syntax

```
New-SmartGroup
  -SamAccountName <string>
  -Name <string>
  -OrganizationalUnit <string>
  -GroupScope <string>
  -Type <string>
  -SecurityType <string>
  [-SearchContainers <string[]>]
  [-SearchContainersScopeList <string[]>]
  [-ObjectTypes <string[]>]
  [-LdapFilter <string>]
  [-IncludeRecipients <string[]>]
  [-ExcludeRecipients <string[]>]
  [-Storage <string>]
  [-DataSourceType <string>]
  [-SystemDSN <string>]
  [-TableorView <string>]
  [-DataSourceUserName <string>]
  [-DataSourcePassword <string>]
  [-FilePath <string>]
  [-Server <string>]
  [-Port <int>]
  [-LDAPSearchContainer <string>]
  [-DataSourceName <string>]
  [-DataSourceConnection <string>]
  [-DataSourceQuery <string>]
  [-KeyMapDB <string>]
  [-KeyMapAD <string>]
  [-WindowsAthentication]
  [-IsPasswordExpiryGroup]
  [-DomainExpiration <int>]
  [-ExpirationRange <int>]
  [-IncludeDisabledUsers <string>]
  [-IncludePasswordNeverExpireUsers <string>]
  [-Script <string>]
  [-ScriptFilePath <string>]
  [-Sun_Container <string>]
  [-GroupAlias <string>]
  [-ManagedBy <string[]>]
  [-DisplayName <string>]
  [-MailEnabled <string>]
  [-Description <string>]
  [-AdditionalOwners <string[]>]
  [-NotifyOptOutAdditionalOwners <string[]>]
  [-Members <string[]>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameters

- SamAccountName
- Name
- OrganizationalUnit
- GroupScope
- Type
- SecurityType

Example 1

The following command creates a new mail-enabled, universal, distribution Smart Group in the container specified by the **OrganizationalUnit** parameter, using the credentials of current user logged-on to the identity store.

```
New-SmartGroup  -OrganizationalUnit
"OU=Recruiting,DC=HR,DC=Imanami,DC=US" -Name
"Smart_Training" -GroupAlias "Smart_Training" -MailEnable
True -SamAccountName "Smart_Training" -GroupScope
"Universal Group" -Type "Distribution"
```

In Microsoft Exchange 2007 and later, mail-enabled groups are created with *Universal* Group Scope.

Example 2

The following command creates a new universal, distribution Smart Group in the container specified by the **OrganizationalUnit** parameter and builds its membership by retrieving those objects from the containers specified in the **SearchContainers** parameter excluding sub-containers whose **Display Names** match the Names in a text file.

```
New-SmartGroup -OrganizationalUnit
"OU=Recruiting,OU=HR,DC=Imanami,DC=US" -Name
"Smart_Enrollment" -SamAccountName "Smart_Enrollment" -
GroupScope "Universal Group" -Type "Distribution" -
SearchContainers
"OU=Recruiting,OU=HR,DC=Imanami,DC=US","OU=Outsourcing,OU=H
R,DC=Imanami,DC=US" -SearchContainersScopeList "1","1" -
LdapFilter "(displayName=Database.[Name])" -DataSourceType
"Microsoft Text Driver (*.txt,*.csv)" -FilePath
"D:\Inputs\Names.txt" -DataSourceQuery "SELECT [Name] FROM
[Names.txt]"
```

Example 3

The following command creates a new local, distribution, Password Expiry group, using the credentials set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A. Those users will be members of the group who have passwords aged 20 days or older. Disabled users will also be included in the membership.

```
New-SmartGroup -OrganizationalUnit
"OU=Recruiting,OU=HR,DC=Imanami,DC=US" -Name
"Password_Expiry" -GroupAlias "Password_Expiry" -
SamAccountName "Password_Expiry" -GroupScope "Domain Local"
-Type "Distribution" -IsPasswordExpiryGroup -
DomainExpiration 30 -ExpirationRange 10 -
IncludeDisabledUsers True -Credential $Cred
```

Set-Options

Use this commandlet to modify the values of GroupID configuration settings in the registry.

Syntax

```
Set-Options
  [-AutomateLoggingLevel <string>]
  [-LoggingLevel <string>]
  [-CleanupApprovedRequests <string>]
  [-CleanupDeniedRequests <string>]
  [-CleanupPendingRequests <string>]
  [-DefaultGroupDeletionTimeAfterExpiry <int>]
  [-DefaultUnusedGroupsExpirationTime <int>]
  [-DefaultExpirationPolicy <int>]
  [-DefaultMaximumNumberOfMembers <int>]
  [-DefaultMaximumNumberOfMembersToDisplay <int>]
  [-DefaultNumberOfOwnersToDisplay <int>]
  [-DefaultReportToMessageOrginator <string>]
  [-DefaultReportToOwner <string>]
  [-DefaultRequestDeletionTime <int>]
  [-DefaultStartWithGlobalCatalogInQueryDesigner <string>]
  [-DeleteEmpty <string>]
  [-DeleteNestedOrphanGroups <string>]
  [-DynastyManagerAsMember <string>]
  [-DeleteExpiredGroups <string>]
  [-ExpireUnusedGroups <string>]
  [-GroupUsageLifecycleEnabled <string>]
  [-DeleteRequests <string>]
  [-EnforceOutOfBounds <string>]
  [-ExcludeOUs  <string>]
  [-ExtensionDataAttributeName <string>]
  [-FirstRun <string>]
```



```
[-FromEmailAddress <string>]
[-GenerateOnedayToExpiryReport <string>]
[-GenerateSevenDaysToExpiryReport <string>]
[-GenerateThirtyDaysToExpiryReport <string>]
[-IncludeExcludeOUs <List[string]>]
[-GroupNamePrefixes <List[string]>]
[-HideMembership <string>]
[-DisplayNestedOwnership <string>]
[-InheritedAttrs <List[string]>]
[-MaximumMembersToDisplay <int>]
[-MaximumOwnersToDisplay <int>]
[-MinimumOwnersToDisplay <int>]
[-NumberOfOwnersToDisplay <int>]
[-OutOfBoundsAlertEnabled <string>]
[-OutOfBoundsMinimum <int>]
[-OutOfBoundsPercentage <int>]
[-PageSize <int>]
[-Prefix <string>]
[-ReportingLoggingLevel <string>]
[-SmartDLNotes <string>]
[-SmtpServer <string>]
[-SelfServiceLoggingLevel <string>]
[-SupportEmail <string>]
[-SupportURL <string>]
[-SynchronizeLoggingLevel <string>]
[-UpdateChildren <string>]
[-DefaultGroupApprover <string>]
[-ConfiguredExchange <int>]
[-EmailProviderDomain <string>]
[-MaximumAliasLength <int>]
[-PasswordCenterSupportURL <string>]
[-KeepHistoryOption <int>]
[-PasswordPortalUrl <string>]
[-SmtpUserName <string>]
[-SmtpPassword <string>]
[-UseSmtpUserAuthentication <string>]
[-SmtpSSLEnabled <string>]
[-SmtpPort <int>]
[-DataServiceURL <string>]
[-SQLServerName <string>]
[-SQLDatabaseName <string>]
[-SQLUserName <string>]
[-SQLPassword <string>]
[-SQLAuthenticationType <int>]
[-WindowsUserName <string>]
[-WindowsPassword <string>]
[-IsSecurityGroupExpirationPluginEnabled <string>]
[-ChangeTrackerActions <List[string]>]
[-GUsExtendGroupsLife <bool>]
[-GUsReduceGroupsLife <bool>]
[-GUsUnusedGroupsTime <int>]
[-GUsusedGroupsTime <int>]
```

```
  [-GLmGroupDeletionInterval <int>]
  [-Delimiter <string>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- None

Example 1

The following command modifies the value of the setting **MaximumMembersToDisplay** to 4000.

```
Set-options -optionnames "MaximumMembersToDisplay" -
OptionValues "4000"
```

Example 2

The following command specifies the SMTP server configurations using the credentials set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A.

```
Set-Options -SmtpServer "smtp.gmail.com" -FromEmailAddress
"ImanamiHR@gmail.com" -UseSmtpUserAuthentication True -
SmtpUserName "ImanamiHR@gmail.com" -SmtpPassword "Abc123*"
-SmtpPort 587 -SmtpSSLEnabled True
```

Example 3

The following command sets multiple values for the setting **InhertiedAttrs**.

```
Set-Options -InheritedAttrs
"ManagedBy","UnauthOrig","DLMemRejectPerms","DLMemSubmitPer
ms","AuthOrig","DelivContLength"
```

Example 4

The following command sets GroupID to track history for **Additional Owner Change**, **Enrollment** and **Expiration Policy Change**.

```
Set-Options -ChangeTrackerActions
"AdditionalOwnerChange#Enrollment#ExpirationPolicyChange"
```

Example 5

The following command enables the group usage lifecycle and set it to reduce the life of distribution groups that have not been sent an e-mail in the last 30 days.

```
Set-Options -GroupUsageLifecycleEnabled True -
ExpireUnusedGroups True -DefaultUnusedGroupsExpirationTime
"30"
```

Set-SmartGroup

The **Set-SmartGroup** commandlet modifies a Smart Group in Directory. Attributes that are common to both Smart Groups and unmanaged groups can also be modified using the Set-Group commandlet.

You can view events related to this commandlet in GroupID Management Console, on the History tab of the object's Properties dialog box.

Syntax

```
Set-SmartGroup
  -Identity <string>
  [-SearchContainers <string[]>]
  [-SearchContainersScopeList <string[]>]
  [-ObjectTypes <string[]>]
  [-LdapFilter <string>]
  [-IncludeRecipients <string[]>]
  [-ExcludeRecipients <string[]>]
  [-Storage <string>]
  [-DataSourceType <string>]
  [-SystemDSN <string>]
  [-TableOrView <string>]
  [-DataSourceUserName <string>]
  [-DataSourcePassword <string>]
  [-FilePath <string>]
  [-Server <string>]
  [-Port <int>]
  [-LDAPSearchContainer <string>]
  [-DataSourceName <string>]
  [-DataSourceQuery <string>]
  [-WindowsAuthentication]
  [-EnableUpdate <string>]
  [-IsPasswordExpirySmartDL]
  [-ExpirationRange <int>]
  [-DomainExpiration <int>]
  [-MaximumPasswordAge <int>]
  [-MinimumPasswordAge <int>]
  [-IncludeDisabledUsers <string>]
  [-IncludePasswordNeverExpireUsers <string>]
```

```
[-SendEmail <string>]
[-EmailTemplatePath <string>]
[-Script <string>]
[-ScriptFilePath <string>]
[-Provider_Container <string>]
[-PowerTools <ArrayList>]
[-KeyMapAD <string>]
[-KeyMapDB <string>]
[-ExtendGroupLife]
[-ExpirationPolicy <int>]
[-MsExchCoManagedByLink <string[]>]
[-IsExpired <string>]
[-GroupScope <string>]
[-Type <string>]
[-Prefix <string>]
[-SecurityType <string>]
[-ManagedBy <string[]>]
[-MaxSendSize <int>]
[-AcceptMessagesOnlyFrom <string[]>]
[-RejectMessagesFrom <string[]>]
[-AcceptMessagesOnlyFromGroups <string[]>]
[-RejectMessagesFromGroup <string[]>]
[-AdditionalOwners <string[]>]
[-NotifyOptOutAdditionalOwners <string[]>]
[-ExpansionServer <string>]
[-BypassOwnersPolicy <string>]
[-MsExchRequireAuthToSendTo <string>]
[-HiddenFromAddressListEnabled <string>]
[-SendOofMessageToOriginatorEnabled <string>]
[-HideMembershipFromAddressListEnabled <string>]
[-ReportToManagerEnabled <string>]
[-ReportToOriginatorEnabled <string>]
[-UpdateMembershipByManagerEnabled <string>]
[-Add <hashtable[]>]
[-Remove <hashtable[]>]
[-Replace <hashtable[]>]
[-Clear <string[]>]
[-Department <string>]
[-Company <string>]
[-Assistant <string>]
[-HomePage <string>]
[-Alias <string>]
[-EmailAddress <string>]
[-Description <string>]
[-Notes <string>]
[-AdministrativeNotes <string>]
[-DisplayName <string>]
[-SimpleDisplayName <string>]
[-CustomAttribute1 <string>]
[-CustomAttribute2 <string>]
[-CustomAttribute3 <string>]
[-CustomAttribute4 <string>]
```

```
  [-CustomAttribute5 <string>]
  [-CustomAttribute6 <string>]
  [-CustomAttribute7 <string>]
  [-CustomAttribute8 <string>]
  [-CustomAttribute9 <string>]
  [-CustomAttribute10 <string>]
  [-CustomAttribute11 <string>]
  [-CustomAttribute12 <string>]
  [-CustomAttribute13 <string>]
  [-CustomAttribute14 <string>]
  [-CustomAttribute15 <string>]
  [-Delimiter <string>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential  <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example 1

The following command modifies a Smart Group by adding **Administrator** statically in the group membership, regardless of whether it is returned by the query, using the credentials of current user logged-on to the identity store.

```
Set-SmartGroup -Identity
"CN=Smart_Training,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -
IncludeRecipients
"CN=Administrator,CN=Users,DC=HR,DC=Imanami,DC=US"
```

Example 2

The following command modifies the LDAP query of a Smart Group to retrieve all mail-enabled objects that are members of the group **Training**, using the credentials set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A.

```
Set-SmartGroup -Identity
"CN=Smart_Training,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -
ObjectTypes
"ExchangeUsers","ExternalUsers","ExternalContacts","EmailGr
oups" -LdapFilter "(MemberOf=Training)" -Credential $Cred
```

Example 3

The following command modifies the Password Expiry group using the credentials of current user logged-on to the identity store. To be added are those users who reside in the containers specified in the **Add** parameter (including sub-containers) and whose password is 20 days or more older and set to never expire.

```
Set-SmartGroup -Identity
"CN=Password_Expiry,OU=Recruiting,OU=HR,DC=Imanami,DC=US" -
Add
@{SearchContainers="OU=Recruiting,OU=HR,DC=Imanami,DC=US#2"
,"OU=Outsourcing,OU=HR,DC=Imanami,DC=US#2" -
IsPasswordExpirySmartDL -DomainExpiration 30 -
ExpirationRange 10 -IncludePasswordNeverExpireUsers True]
```

Example 4

The following command modifies the membership of a Smart Group based on the script given in the script file.

```
Set-SmartGroup -Identity
"CN=Smart_Training,OU=Recruiting,DC=HR,DC=Imanami,DC=US" –
ScriptFilePath "c:\MembershipUpdateScript.vb"
```

Example 5

The following command overwrites the **Includes** and **Excludes** lists of a Smart Group by adding two groups in the Includes list and one group in the Excludes list.

```
Set-SmartGroup -Identity "CN=imrantest, OU=Testit,
DC=minion,DC=local" –Replace @{Includes =
"CN=Shizasss,CN=Users,DC=minion,DC=Local","CN=ShezaOfc,CN=U
sers,DC=minion,DC=Local" ;
Excludes="CN=Administrator,CN=Users,DC=minion,DC=local" ,
"CN=TestMailbox,CN=Users,DC=minion,DC=local"]
```

Exampe 6

The following command modifies lists of members a Smart Group can accept and reject messages from.

```
Set-SmartGroup -Identity
"CN=Smart_Training,OU=Recruiting,DC=HR,DC=Imanami,DC=US" –
Add @{ RejectMessagesFrom =
"CN=Roger_Manson,OU=ResignedStaff,DC=HR,DC=Imanami,DC=US"}
-Add @(AcceptMessageOnlyFrom =
"CN=PKWing,OU=Recruiting,DC=HR,DC=Imanami,DC=US","CN=USWing
,OU=Recruiting,DC=HR,DC=Imanami,DC=US")
```


Update-Group

The **Update-Group** commandlet modifies the membership of a Smart Group or Dynasty according to the results returned by the LDAP query. This query is associated with the group or Dynasty creation and can be updated anytime using the **Set-SmartGroup** commandlet. When the **Update-Group** commandlet is executed, it searches the directory to find recipients matching the criteria defined in the query and modifies the group membership list with the returned recipients, if any.

You can view events related to this commandlet in GroupID Management Console, on the History tab of the object's Properties dialog box.

Syntax

```
Update-Group
  -Identity <string>
  [-SearchContainer <string>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example 1

The following command updates all the GroupID group(s), by using the credentials of a locally logged on user, in a container specified by the "SearchContainer" parameter.

```
Update-Group –SearchContainer “OU=Sales,DC=Contoso,DC=com”
```

Example 2

The following command updates all Smart Groups and Dynasties present in the container **Training**, using the credentials set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A.

```
Update-Group -SearchContainer
"OU=Training,DC=HR,DC=Imanami,DC=US" -Credential $Cred
```

Upgrade-Group

Upgrade-Group commandlet upgrades managed (Smart Groups and Dynasties) and non-managed Groups of GroupID 7, 8 and 9 to GroupID 10.0 version.

GroupID upgrades groups from the connected database to the current instance of GroupID. This database can be an upgraded version or copied database from the previous GroupID versions i.e. GroupID 7, 8 and 9.

Syntax

```
Upgrade-Group
  -SQLServer <string>
  -Database <string>
  -SQLUserName <string>
  -Password <string>
  -GroupIDVersion <int>
  [-SearchContainer <List[string]>]
  [-SearchContainerScopeList <List[int]>]
  [-Identity <List[string]>]
  [-GroupType <List[int]>]
  [-KeepUserHistory]
  [-ExtensionDataAttributes <List[string]>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameters

- SQLServer
- Database
- SQLUserName
- Password
- GroupIDVersion

Example 1

The following command upgrades a GroupID 7.0 Smart Group *GIDSmart1* using the database *GroupID7SR1* which resides on SQL server *sqlexpress*. To upgrade the smart group to GroupID 10.0 version, the command uses ***sa*** user account of the specified SQL server.

```
Upgrade-Group -Identity "GIDsmart1" -SQLServer
"msvr02\sqlexpress" -SQLUserName "sa" -Database
"GroupID7SR1" -Password "support123R" -GroupIDVersion "7.0"
-GroupType "2"
```

Example 2

The following command upgrades all GroupID 7.0 Smart Groups in *AutomateJobs* container using *GroupID7SR1* database which resides on SQL server *sqlexpress*. To upgrade the smart groups to GroupID 10.0 version, the command uses ***sa*** user account of the specified SQL server.

```
Upgrade-Group -SearchContainer
"OU=AutomateJobs,DC=Demo1,DC=com" -SQLServer
"msvr02\sqlexpress" -SQLUserName "sa" -Database
"GroupID7SR1" -Password "support123R" -GroupIDVersion "7.0"
-GroupType "2"
```

Example 3

The following command upgrades all GroupID 7.0 dynasties in *AutomateJobs* container using GroupID7SR1 database which resides on SQL server *sqlexpress*. To upgrade the dynasties to GroupID 10.0 version, the command uses ***sa*** user account of the specified SQL server.

```
Upgrade-Group -SearchContainer
"OU=AutomateJobs,DC=Demo1,DC=com" -SQLServer
"msvr02\sqlexpress" -SQLUserName "sa" -Database
"GroupID7SR1" -Password "support123R" -GroupIDVersion "7.0"
-GroupType "3"
```

Example 4

The following command upgrades non managed groups in *GID7* container using GroupID7SR1 database which resides on SQL server *sqlexpress*. To upgrade the non-managed groups to GroupID 10.0 version, the command uses ***sa*** user account of the specified SQL server.

```
Upgrade-Group -Identity "departsales" -SearchContainer
"OU=GID7,DC=Demo1,DC=com" -SQLServer "msvr02\sqlexpress" -
SQLUserName "sa" -Database "GroupID7SR1" -Password
"support123R" -GroupIDVersion "7.0" -GroupType "1"
```

Chapter 10 - Dynasty Commands

This chapter covers commandlets for managing dynasties.

- New-Dynasty: creates a new dynasty.
- Set-Dynasty: modifies a Dynasty or its children.

Review the description of the supported parameters of these commandlets along with their attributes in the List of Parameters**Error! Reference source not found.** table.

Common parameters of Windows Management Shell are not supported in GroupID Management Shell.

New-Dynasty

The **New-Dynasty** commandlet creates a new Dynasty in Directory. A Dynasty is a Smart Group that can create and maintain the membership of other Smart Groups. A Dynasty retrieves data from Directory in the same manner as a Smart Group, but it divides the result set into child groups based on group-by field values.

You can specify multiple group-by fields. For instance, with the group-by fields Country, State, and City, this commandlet creates a group for every distinct country value, then for each state within a country, and finally for each city in that state. All created child groups inherit those attributes of the parent that are set in the **InheritedAttrs** option. This attribute list can be viewed using the Get-Options commandlet.

You can view events related to this commandlet in GroupID Management Console, on the History tab of the object's Properties dialog box.

Syntax

```
New-Dynasty
  -TopManager <string>
  -SamAccountName <string>
  -Name <string>
  -OrganizationalUnit <string>
  -GroupScope <string>
  -Type <string>
  -SecurityType <string>
  [-ChildContainer <string[]>]
  [-Filters <string[]>]
```

```
  [-Separator <string[]>]
  [-ExcludeNestedLists <string>]
  [-CreateFlatManagerialList <string>]
  [-IncludeManagerAsMember <string>]
  [-ChildPath <string>]
  [-DynastyInheritance <bool>]
  [-SearchContainers <string[]>]
  [-SearchContainersScopeList <string[]>]
  [-ObjectTypes <string[]>]
  [-LdapFilter <string>]
  [-IncludeRecipients <string[]>]
  [-ExcludeRecipients <string[]>]
  [-Storage <string>]
  [-DataSourceType <string>]
  [-SystemDSN <string>]
  [-TableorView <string>]
  [-DataSourceUserName <string>]
  [-DataSourcePassword <string>]
  [-FilePath <string>]
  [-Server <string>]
  [-Port <int>]
  [-LDAPSearchContainer <string>]
  [-DataSourceName <string>]
  [-DataSourceConnection <string>]
  [-DataSourceQuery <string>]
  [-KeyMapDB <string>]
  [-KeyMapAD <string>]
  [-WindowsAthentication]
  [-IsPasswordExpiryGroup]
  [-DomainExpiration <int>]
  [-ExpirationRange <int>]
  [-IncludeDisabledUsers <string>]
  [-IncludePasswordNeverExpireUsers <string>]
  [-Script <string>]
  [-ScriptFilePath <string>]
  [-Sun_Container <string>]
  [-GroupAlias <string>]
  [-ManagedBy <string[]>]
  [-DisplayName <string>]
  [-MailEnabled <string>]
  [-Description <string>]
  [-AdditionalOwners <string[]>]
  [-NotifyOptOutAdditionalOwners <string[]>]
  [-Members <string[]>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameters

- TopManager
- SamAccountName
- Name
- OrganizationalUnit
- GroupScope
- Type
- SecurityType

Example 1

The following command creates a new mail-enabled, universal, distribution Dynasty and constructs its child groups for every distinct department value in the container specified by the **OrganizationalUnit** parameter using the credentials of current user logged-on to the identity store.

```
New-Dynasty -OrganizationalUnit
"OU=Recruiting,DC=HR,DC=Imanami,DC=US" -Name "Departmental"
-SamAccountName  "Departmental" -Type "Distribution" -
GroupScope "Universal Group" -MailEnable True -GroupAlias
"Departmental" -GroupBy  "Department"
```

Example 2

The following command creates a new mail-enabled, universal, distribution, multi-level Dynasty with the group-by attributes **Country**, **State** and **City** based on the specified filters and separator, using the credentials set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A.

```
New-Dynasty -OrganizationalUnit
"OU=Recruiting,DC=HR,DC=Imanami,DC=US" -Name "Geographical"
-GroupAlias "Geographical" -MailEnable True -SamAccountName
"Geographical" -GroupScope "Universal Group" -Type
"Distribution" -GroupBy "co","st","l" -Filters "Left
3","Left 3","%GROUPBY%\*" -Separator "_","_","_" -
Credential $Cred
```

Example 3

The following command creates a new universal, distribution Managerial Dynasty in the container specified by the **OrganizationalUnit** parameter, searches the direct reports of the top manager in the containers specified in the **SearchContainers** parameter including sub containers and creates them in the same container where the Top Manager resides.

```
New-Dynasty -OrganizationalUnit
"OU=Recruiting,DC=HR,DC=Imanami,DC=US" -Name "Managerial" -
SamAccountName "Managerial" -GroupScope "Universal Group" -
Type "Distribution" -SearchContainers
"OU=Recruiting,OU=HR,DC=Imanami,DC=US","OU=Outsourcing,OU=H
R,DC=Imanami,DC=US" -SearchContainersScopeList "2","2" -
TopManager "CN=BrianRegan,CN=Users,DC=HR,DC=Imanami,DC=US"
-ExcludeNestedLists False -ChildContainer ""
```

Set-Dynasty

The **Set-Dynasty** commandlet lets you to modify a Dynasty or its children in Directory.

GroupID maintains a history for this commandlet, which you can view in GroupID Management Console using the History tab of the object's properties dialog box.

Syntax

```
Set-Dynasty
  -Identity <string>
  [-GroupBy <string[]>]
  [-AliasTemplate <string>]
  [-DisplayNameTemplate <string>]
  [-InheritanceBehaviour
   {InheritSelectedAttributeOnCreation |
    AlwaysInheritSelectedAttributes |
    NeverInheritSelectedAttributes}]
  [-TopManager <string>]
  [-ChildContainer <string[]>]
  [-ExcludeNestedLists <string>]
  [-CreateFlatManagerialList <string>]
  [-IncludeManagerAsMember <string>]
  [-Filters <string[]>]
  [-Separator <string[]>]
  [-SearchContainers <string[]>]
  [-SearchContainersScopeList <string[]>]
  [-ObjectTypes <string[]>]
  [-LdapFilter <string>]
  [-IncludeRecipients <string[]>]
  [-ExcludeRecipients <string[]>]
  [-Storage <string>]
  [-DataSourceType <string>]
  [-SystemDSN <string>]
  [-TableOrView <string>]
  [-DataSourceUserName <string>]
  [-DataSourcePassword <string>]
  [-FilePath <string>]
  [-Server <string>]
```

```
  [-Port <int>]
  [-LDAPSearchContainer <string>]
  [-DataSourceName <string>]
  [-DataSourceQuery <string>]
  [-WindowsAuthentication]
  [-EnableUpdate <string>]
  [-IsPasswordExpirySmartDL]
  [-ExpirationRange <int>]
  [-DomainExpiration <int>]
  [-MaximumPasswordAge <int>]
  [-MinimumPasswordAge <int>]
  [-IncludeDisabledUsers <string>]
  [-IncludePasswordNeverExpireUsers <string>]
  [-SendEmail <string>]
  [-EmailTemplatePath <string>]
  [-Script <string>]
  [-ScriptFilePath <string>]
  [-Provider_Container <string>]
  [-PowerTools <ArrayList>]
  [-KeyMapAD <string>]
  [-KeyMapDB <string>]
  [-ExtendGroupLife]
  [-ExpirationPolicy <int>]
  [-MsExchCoManagedByLink <string[]>]
  [-IsExpired <string>]
  [-GroupScope <string>]
  [-Type <string>]
  [-Prefix <string>]
  [-SecurityType <string>]
  [-ManagedBy <string[]>]
  [-MaxSendSize <int>]
  [-AcceptMessagesOnlyFrom <string[]>]
  [-RejectMessagesFrom <string[]>]
  [-AcceptMessagesOnlyFromGroups <string[]>]
  [-RejectMessagesFromGroup <string[]>]
  [-AdditionalOwners <string[]>]
  [-NotifyOptOutAdditionalOwners <string[]>]
  [-ExpansionServer <string>]
  [-BypassOwnersPolicy <string>]
  [-MsExchRequireAuthToSendTo <string>]
  [-HiddenFromAddressListEnabled <string>]
  [-SendOofMessageToOriginatorEnabled <string>]
  [-HideMembershipFromAddressListEnabled <string>]
  [-ReportToManagerEnabled <string>]
  [-ReportToOriginatorEnabled <string>]
  [-UpdateMembershipByManagerEnabled <string>]
  [-Add <hashtable[]>]
  [-Remove <hashtable[]>]
  [-Replace <hashtable[]>]
  [-Clear <string[]>]
  [-Department <string>]
  [-Company <string>]
```

```
  [-Assistant <string>]
  [-HomePage <string>]
  [-Alias <string>]
  [-EmailAddress <string>]
  [-Description <string>]
  [-Notes <string>]
  [-AdministrativeNotes <string>]
  [-DisplayName <string>]
  [-SimpleDisplayName <string>]
  [-CustomAttribute1 <string>]
  [-CustomAttribute2 <string>]
  [-CustomAttribute3 <string>]
  [-CustomAttribute4 <string>]
  [-CustomAttribute5 <string>]
  [-CustomAttribute6 <string>]
  [-CustomAttribute7 <string>]
  [-CustomAttribute8 <string>]
  [-CustomAttribute9 <string>]
  [-CustomAttribute10 <string>]
  [-CustomAttribute11 <string>]
  [-CustomAttribute12 <string>]
  [-CustomAttribute13 <string>]
  [-CustomAttribute14 <string>]
  [-CustomAttribute15 <string>]
  [-Delimiter <string>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example 1

The following command modifies the **Departmental** Dynasty by changing the Group-by attributes list using the credentials of current user logged-on to the identity store.

```
Set-Dynasty -Identity
"CN=DepartmentalOU=Recruiting,DC=HR,DC=Imanami,DC=US" -
GroupBy "Department","Company","Title"
```

Example 2

The command below modifies the **Top Manager** of a Managerial Dynasty, changes the alias name and display name templates for the Dynasty children, sets the scope to search Dynasty children in the containers specified in the **Add** parameter excluding sub-containers using the credentials set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A.


```
Set-Dynasty -Identity
"CN=Managerial,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -
TopManager
"CN=Administrator,CN=Users,DC=HR,DC=Imanami,DC=US" -Add @{
SearchContainers="OU=Recruiting,OU=HR,DC=Imanami,DC=US#1","
OU=Outsourcing,OU=HR,DC=Imanami,DC=US#1"}-
ExcludeNestedLists False -ChildContainer "" -AliasTemplate
"%Manager% -DirectReports" -DisplayNameTemplate "Direct
reports of %Manager%" -Credential $Cred
```

Example 3

The following command modifies the search criteria for the Managerial Dynasty to retrieve all mail-enabled objects who are the member of the **Training** group.

```
Set-Dynasty -Identity
"CN=Managerial,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -
ObjectTypes
"ExchangeUsers","ExternalUsers","ExternalContacts","EmailGr
oups" -LdapFilter "(MemberOf=Training)"
```

Example 4

The following command adds three group-by levels to an Organizational Dynasty.

```
Set-Dynasty -Identity
"CN=Organizational,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -
Add
@{GroupBy="Company#OU=Recruiting,DC=HR,DC=Imanami,DC=US#Lef
t 3#-
","Department#OU=Recruiting,DC=HR,DC=Imanami,DC=US#Right
5#-","OU=Recruiting,DC=HR,DC=Imanami,DC=US#With
%GROUPBY%\*#^"}
```

Example 5

The following command modifies additional owners, **Includes** and **Excludes** lists and replaces Search Scope of a Managerial Dynasty.

```
Set- Dynasty -Identity
"CN=Managerial,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -Add
@{AdditionalOwners="CN=Roger
Manson,OU=Recruiting,OU=HR,DC=Imanami,DC=US","Robin Soto";
Includes="USWing","PKWing"; Excludes="UAEWing"} -Replace
@{SearchContainers="OU=Recruiting,OU=HR,DC=Imanami,DC=US#1"
,"OU=Outsourcing,OU=HR,DC=Imanami,DC=US#1"}
```

Example 6

The following command clears the groups specified in the **Includes** list of a Managerial Dynasty.

```
Set-Dynasty -Identity
"CN=Managerial,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -Clear
"Includes"
```


Chapter 11 - Mailbox Commands

This chapter covers commandlets for performing mailbox related tasks such as:

- Get-Mailbox: retrieves a mailbox.
- New-Mailbox: creates a new mailbox.
- Remove-Mailbox: deletes a mailbox.
- Set-Mailbox: modifies a mailbox.

Review the description of the supported parameters of these commandlets along with their attributes in the List of Parameters**Error! Reference source not found.** table.

Common parameters of Windows Management Shell are not supported in GroupID Management Shell.

Get-Mailbox

Use the **Get-Mailbox** commandlet to retrieve basic information about a mailbox that match your given criteria.

Syntax

```
Get-MailBox
   [[-Identity] <string[]>]
   [-SearchContainer <string[]>]
   [-SearchContainersScopeList <string>]
   [-MailBoxStore <string>]
   [-ShouldReturnCollection]
   [-MaxItemsToDisplay <int>]
   [-ObjectType <string[]>]
   [-LdapFilter <string>]
   [-SmartFilter <string>]
   [-ServerFilter <string>]
   [-AttributesToLoad <string[]>]
   [-IdentityStoreId <int>]
   [-SecurityToken <CustomClaimsPrincipal>]
   [-Credential <pscredential>]
   [<CommonParameters>]
```

Required parameter

- None

Example

The following command retrieves the specified mailbox from the connected identity store.

```
Get-MailBox -Identity "OsamaMailBox"
```

New-Mailbox

Use the **New-Mailbox** commandlet to create a new mailbox in Directory. Most mailbox properties can be directly added by using the parameters of this commandlet.

Syntax

```
New-MailBox
  -MailBoxStore <string>
  -Alias <string>
  -Name <string>
  -OrganizationalUnit <string>
  -SAMAccountName <string>
  -Password <string>
  -FirstName <string>
  -LastName <string>
  -DisplayName <string>
  [-UPNSuffix <string>]
  [-Title <string>]
  [-City <string>]
  [-State <string>]
  [-Zip <string>]
  [-Country <string>]
  [-Initials <string>]
  [-Address <string>]
  [-Office <string>]
  [-Business <string>]
  [-Business2 <string>]
  [-EmailAddress <string>]
  [-Department <string>]
  [-Company <string>]
  [-Mobile <string>]
  [-Home <string>]
  [-AccountDisabled <string>]
  [-PasswordNeverExpires <string>]
  [-PasswordForceChange <string>]
  [-Manager <string[]>]
```



```
  [-HomePage <string>]
  [-Assistant <string>]
  [-Notes <string>]
  [-MailEnabled <string>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameters

- MailBoxStore
- Alias
- Name
- OrganizationalUnit
- SAMAccountName
- Password
- FirstName
- LastName
- DisplayName

Example

The following command creates a new mailbox in the container specified by the **OrganizationalUnit** parameter of specified mailbox store. The command also specifies the logon name, password, first name, last name and display name of the new mailbox.

```
New-MailBox -MailBoxStore "OsamaMailBoxDb120435" -Name
"OsamaMailBox" -OrganizationalUnit
"OU=osamamu,DC=naveed,DC=local" -SAMAccountName
"OsamaMailBoxUser" -Password "webdir123R" -FirstName
"OsamaMailBox" -LastName "MailBoxuser" -DisplayName
"OsamaMailBox" -Alias "OsamaMailBox
```

Remove-Mailbox

Use the **Remove-Mailbox** commandlet to delete mailbox from the connected identifty store.

Syntax

```
Remove-MailBox
  -Identity <string[]>
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example

The following command deletes the specified mailbox from the connected identity store.

```
Remove-MailBox -Identity "OsamaMailBox"
```

Set-Mailbox

The **Set-Mailbox** commandlet modifies a mailbox in Directory. Most mailbox properties can be directly modified by using the parameters of this commandlet.

Syntax

```
Set-MailBox
  -Identity <string>
  [-FirstName <string>]
  [-LastName <string>]
  [-Title <string>]
  [-City <string>]
  [-State <string>]
  [-Zip <string>]
  [-Country <string>]
  [-Initials <string>]
  [-Address <string>]
  [-Office <string>]
  [-Business <string>]
  [-Add <hashtable[]>]
  [-Remove <hashtable[]>]
  [-Replace <hashtable[]>]
  [-Clear <string[]>]
  [-Department <string>]
  [-Company <string>]
  [-Assistant <string>]
  [-HomePage <string>]
  [-Alias <string>]
  [-EmailAddress <string>]
```



```
  [-Description <string>]
  [-Notes <string>]
  [-AdministrativeNotes <string>]
  [-DisplayName <string>]
  [-SimpleDisplayName <string>]
  [-CustomAttribute1 <string>]
  [-CustomAttribute2 <string>]
  [-CustomAttribute3 <string>]
  [-CustomAttribute4 <string>]
  [-CustomAttribute5 <string>]
  [-CustomAttribute6   <string>]
  [-CustomAttribute7 <string>]
  [-CustomAttribute8 <string>]
  [-CustomAttribute9 <string>]
  [-CustomAttribute10 <string>]
  [-CustomAttribute11 <string>]
  [-CustomAttribute12 <string>]
  [-CustomAttribute13 <string>]
  [-CustomAttribute14 <string>]
  [-CustomAttribute15 <string>]
  [-Delimiter <string>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example

The following commandlet modifies the country value of the specified mailbox in the connected identity store.

```
Set-MailBox -Identity "OsamaMailBox" -Country "Pakistan"
```


Chapter 12 - Mail-Enabled/Disabled Groups Commands

This chapter covers commandlets for enabling and disabling groups for email.

- Disable-DistributionGroup: disabled a group's email capability for a group.
- Enable-DistributionGroup: enable a group's email capability for a group.

Review the description of the supported parameters of these commandlets along with their attributes in the List of Parameters**Error! Reference source not found.** table.

Common parameters of Windows Management Shell are not supported in GroupID Management Shell.

Disable-DistributionGroup

Use this commandlet to disable the mailing capabilities for a distribution group in Directory.

GroupID maintains a history for this commandlet, which you can view in GroupID Management Console using the History tab of the object's properties dialog box.

Syntax

```
Disable-DistributionGroup
  -Identity <string>
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example

The following command mail-disables a distribution group specified by the **Identity** parameter, using the credentials of current user logged-on to the identity store.

```
Disable-DistributionGroup -Identity
"CN=Smart_Training,OU=Recruiting,DC=HR,DC=Imanami,DC=US"
```

Enable-DistributionGroup

This commandlet helps you to mail-enable a distribution group in Directory.

GroupID maintains a history for this commandlet, which you can view in GroupID Management Console using the History tab of the object's properties dialog box.

Syntax

```
Enable-DistributionGroup
  -Identity <string>
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example

The following command mail-enables a distribution group specified by the **Identity** parameter, using the credentials of current user logged-on to the identity store.

```
Enable-DistributionGroup -Identity
"CN=Smart_Training,OU=Recruiting,DC=HR,DC=Imanami,DC=US"
```


Chapter 13 - Memberships Commands

This chapter covers commandlets for managing the memberships of both managed and unmanaged groups.

- Add-GroupMember: adds objects to the membership of a group.
- Get-GroupMember: retrieves members of a particular group.
- Get-Object: retrieves objects.
- Remove-GroupMemeber: removes recipients from a group membership.
- Set-Object: modifies any object.

Review the description of the supported parameters of these commandlets along with their attributes in the List of Parameters**Error! Reference source not found.** table.

Common parameters of Windows Management Shell are not supported in GroupID Management Shell.

Add-GroupMember

The **Add-GroupMember** commandlet helps you to add one or more objects to the membership of a group in Directory. Two types of membership can exist in the GroupID.

- Perpetual membership
- Temporary membership

Modifying the membership of a Smart Group or Dynasty using this commandlet is not recommended, since your changes will be discarded the next time the group is updated.

GroupID maintains a history for this commandlet, which you can view in GroupID Management Console using the History tab of the object's properties dialog box.

Syntax

```
Add-GroupMember
  -GroupIdentity <string>
  -Identity <string>
  [-Type <string>]
  [-StartDate <datetime>]
  [-EndDate <datetime>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameters

- GroupIdentity
- Identity

Example 1

The following command adds the user **Brian Regan** to the membership of the **Event Management** group using the credentials set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A.

```
Add-GroupMember -GroupIdentity "CN=Event
Management,OU=Local
Recruiting,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -Identity
"CN=BrianRegan,CN=User,DC=HR,DC=Imanami,DC=US" -Credential
$Cred
```

Example 2

The following command gets all users from the **Local Recruiting** container and adds them to the membership of the **Event Management** group. For detailed information about the Get-Object commandlet, see Get-Object. The **OUT-NULL** commandlet is used here to restrict the retrieved users information from appearing on the console.

```
Get-Object -SearchContainer "OU=Local
Recruiting,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -
ObjectType "User" | Add-GroupMember -GroupIdentity
"CN=Event Management,OU=Local
Recruiting,OU=Recruiting,DC=HR,DC=Imanami,DC=US"
```

Get-GroupMember

Use this commandlet to retrieve members of a particular group from directory. You can apply filters to the results returned by the commandlet.

Syntax

```
Get-GroupMember
  [-Identity] <string>
  [[-LdapFilter] <string>]
  [-AttributesToLoad <string[]>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- None

Example 1

The following command retrieves all members of the **Password_Expiry** group using the credentials set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A.

```
Get-GroupMember -Identity
"CN=Password_Expiry,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -
Credential $Cred
```

Example 2

The command below retrieves all members from the **Enrollment** group whose display name starts with the character **S** using the credentials of current user logged-on to the identity store.

```
Get-GroupMember -Identity "CN=Enrollment,OU=Local
Recruiting,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -
LdapFilter "(displayname=S*)"
```

Get-Object

Use this commandlet to retrieve objects from one or more containers in a domain that match the given criteria.

Syntax

```
Get-Object
  [[-Identity] <string[]>]
  [-ShouldReturnCollection]
  [-MaxItemsToDisplay <int>]
  [-ObjectType <string[]>]
  [-SearchContainer <string[]>]
  [-SearchContainersScopeList <string>]
  [-LdapFilter <string>]
  [-SmartFilter <string>]
  [-ServerFilter <string>]
  [-AttributesToLoad <string[]>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameter

- None

Example 1

The following command retrieves all objects from the domain you are connected to.

```
Get-Object
```

Example 2

The command below retrieves the object **Event Management** starting from the container **Recruiting** excluding its sub-containers using the credentials set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A.

```
Get-Object -Identity "HR.Imanami.US\Event Management" -
SearchContainer "OU=Recruiting,DC=HR,DC=Imanami,DC=US" -
SearchContainersScopeList "1" -Credential $Cred
```

Example 3

The following command searches all objects in the specified containers including sub-containers with display names starting with the letter **S** .

```
Get-Object -SearchContainer
"OU=Recruiting,DC=HR,DC=Imanami,DC=US","OU=OutSourcing,DC=H
R,DC=Imanami,DC=US" -LdapFilter "(DisplayName = S*)"
```


Remove-GroupMember

Use this commandlet to remove one or more recipients from a group membership.

GroupID maintains a history for this commandlet, which you can view in GroupID Management Console using the History tab of the object's properties dialog box.

Syntax

```
Remove-GroupMember
  -GroupIdentity <string>
  -Identity <string>
  [-Type <string>]
  [-StartDate <datetime>]
  [-EndDate <datetime>]
  [-IdentityStoreId <int>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <pscredential>]
  [<CommonParameters>]
```

Required parameters

- GroupIdentity
- Identity

Example

The following command removes the user **Brian Regan** from the membership of the group **Event Management** using the credentials set in the **$Credentials** environment variable. For information about setting credentials, see Appendix A.

```
Remove-GroupMember -GroupIdentity "CN=Event
Management,OU=Local
Recruiting,OU=Recruiting,DC=HR,DC=Imanami,DC=US" -Identity
"Brian Regan" -Credential $Cred
```

Set-Object

The **Set-Object** commandlet modifies any object such as a user, contact, group (managed or unmanaged), or mailbox in Directory.

Syntax

```
Set-Object
  -Identity <String>
  [-Department <String>]
  [-Company <String>]
```



```
  [-Assistant <String>]
  [-HomePage <String>]
  [-Alias <String>]
  [-EmailAddress <String>]
  [-Description <String>]
  [-Notes <String>]
  [-AdministrativeNotes <String>]
  [-DisplayName <String>]
  [-SimpleDisplayName <String>]
  [-CustomAttribute1 <String>]
  [-CustomAttribute2 <String>]
  [-CustomAttribute3 <String>]
  [-CustomAttribute4 <String>]
  [-CustomAttribute5 <String>]
  [-CustomAttribute6 <String>]
  [-CustomAttribute7 <String>]
  [-CustomAttribute8 <String>]
  [-CustomAttribute9 <String>]
  [-CustomAttribute10 <String>]
  [-CustomAttribute11 <String>]
  [-CustomAttribute12 <String>]
  [-CustomAttribute13 <String>]
  [-CustomAttribute14 <String>]
  [-CustomAttribute15 <String>]
  [-Delimiter <String>]
  [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-Credential <PSCredential>]
  [<CommonParameters>]
```

Required parameter

- Identity

Example 1

The following example modifies description of a user specified against the Identity parameter.

```
Set-object -identity "Sonia Iqbal" -Description TestUser
```


Chapter 14 - Scheduling Commands

This section contains the references to the commandlets dealing with the scheduling operations. These commandlets comprise the following set:

- Get-Schedule: retrieves scheduled jobs.
- Get-TargetSchedules: retrieves the scheduled jobs operating on the specified groups or OU.
- Invoke-Schedule: executes the specified scheduled job.
- New-Schedule: creates a new schedule.
- Remove-Schedule: removes a schedule from an identity store.
- Set-Schedule: modifies a schedule.
- Stop-Schedule: stops a specified schedule, if running.

Get-Schedule

The commandlet **Get-Schedule** retrieves the scheduled jobs created in the identity store connected to the current instance of the Management Shell. By default, this cmdlet returns all the jobs available irrespective of the following:

- whether the identity store with which they belong is enabled.
- whether the jobs are enabled.

This commandlet can also filter the job list if provided with the filtration parameters such as JobType, TriggerType or HavingNotifications. It also accepts a MatchingCriteria parameter that determines whether the criteria are to be joined on the AND basis or OR basis.

Syntax

```
Get-Schedule [-ScheduleNames <String[]>]
  [-IdentityStoreNames <String[]>]
  [-JobTypes <JobType[]>]
  [-TriggerTypes <TriggerType[]>]
  [-HavingNotifications <Boolean>]
  [-MatchingCriteria <JoiningOperator>]
  [-PreventEnumeration]
  [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-WarningAction <ActionPreference>]
  [-InformationAction <ActionPreference>]
  [-WarningVariable <String>]
  [-InformationVariable <String>]
  [-PipelineVariable <String>]
  [<CommonParameters>]
```

Required parameter

- None

Example 1

This example retrieves all the scheduled jobs created in the connected identity store.

```
Get-Schedule
```

Example 2

This example retrieves those Group Usage Service – *GUS* job(s) that have monthly trigger and MatchingCriteria on the *And* basis.

```
Get-Schedule -JobType GUS -TriggerType RunMonthly -
MatchingCriteria And
```

Example 3

This example retrieves the scheduled job with *GUS1* name.

```
Get-Schedule -ScheduleName GUS1
```

Example 4

This example retrieves the two scheduled jobs – *GUS1* and *GLM6* –through the pipeline operator.

```
'GUS1','GLM6' | Get-Schedule
```


Get-TargetSchedules

The commandlet **Get-TargetSchedules** retrieves the scheduled jobs operating on the given target (group/OU).

Syntax

```
Get-TargetSchedules
  [-DistinguishedName] <String>
  [-Enumerate]
  [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-WarningAction <ActionPreference>]
  [-InformationAction <ActionPreference>]
  [-WarningVariable <String>]
  [-InformationVariable <String>]
  [-PipelineVariable <String>]
  [<CommonParameters>]
```

Required parameter

- DistinguishedName

Example 1

This example retrieves the schedules operating on an OU with distinguished name OU=WorkingOU,DC=pucit,DC=local.

```
Get-TargetSchedules -DistinguishedName
‘OU=WorkingOU,DC=pucit,DC=local’
```

Example 2

This example retrieves the schedules operating on a group and an OU through the pipeline operator.

```
'OU=WorkingOU,DC=pucit,DC=local',
'CN=SGroup1,OU=ArslanAhmadOU,OU=WorkingOU,DC=pucit,DC=local
' | Get-TargetSchedules
```

Example 3

This example selects only the Names and Job Types of the schedules operating on the specified targets through the pipeline operator.

```
'OU=WorkingOU,DC=pucit,DC=local',
'CN=SGroup1,OU=ArslanAhmadOU,OU=WorkingOU,DC=pucit,DC=local
' | Get-TargetSchedules | Select-Object -Property
Name,JobType
```


Invoke-Schedule

The commandlet **Invoke-Schedule** executes the specified schedules job.

Syntax

```
Invoke-Schedule
  [-ScheduleName <String>]
  [-JobId <Int32>]
  [-PassThru]
  [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-WarningAction <ActionPreference>]
  [-InformationAction <ActionPreference>]
  [-WarningVariable <String>]
  [-InformationVariable <String>]
  [-PipelineVariable <String>]
  [<CommonParameters>]
```

Required parameter

- None

Example 1

This example executes a schedule with name starting smm4_.

```
Invoke-Schedule -ScheduleName smm4_
```

Example 2

This example executes a schedule with GUS as Job Type.

```
Get-Schedule -JobType GUS | Select-Object -Property Name |
Invoke-Schedule
```

Example 3

This example executes all the GUS scheduled jobs with daily running trigger.

```
Get-Schedule -JobType GUS -TriggerType RunDaily -
MatchingCriteria And | Select-Object -Property Name |
Invoke-Schedule
```


New-Schedule

The commandlet **New-Schedule** creates a new schedule in the identity store connected to the current instance of Management Shell.

Syntax

```
New-Schedule
  -ScheduleName <String>
  -Targets <String[]>
  -TargetType <SchedulingTargetType>
  -IdentityStoreName <String>
  -Credentials <PSCredential>
  -JobType <JobType>
  -TriggerType <TriggerType>
  -StartTime <DateTime>
  [-WeekDays <DaysOfTheWeek>]
  [-YearMonths <MonthsOfTheYear>]
  [-MonthDate <Int32>]
  [-EnableNotifications]
  [-Recepients <String[]>]
  [-SendToOwners]
  [-NotificationSendingCriteria
<NotificationSendingCriteria>]
  [-PassThru] [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-WarningAction <ActionPreference>]
  [-InformationAction <ActionPreference>]
  [-WarningVariable <String>]
  [-InformationVariable <String>]
  [-PipelineVariable <String>]
  [<CommonParameters>]
```

Required parameters

- ScheduleName
- Targets
- TargetType
- IdentityStoreName
- Credentials
- JobType
- TriggerType
- StartTime

Example 1

This example creates a new schedule using minimum possible parameters. This example contains insecure password.

```
New-Schedule -ScheduleName SmuTest1 -IdentityStoreName
AdStore8 -UserName user -Password password1 -Targets
'OU=ArslanAhmadOU,OU=WorkingOU,DC=pucit,DC=local',
'OU=ArslanAhmadOU,OU=WorkingOU,DC=pucit,DC=local' -JobType
SmartGroup -TriggerType Daily -StartTime '16:56'
```

This example uses insecure credentials.

Example 2

This example creates a smart-group schedule triggering every 7th of every March, August and September.

```
New-Schedule -ScheduleName SmuTest2 -IdentityStoreName
AdStore8 -Credentials $creds -Targets
'OU=ArslanAhmadOU,OU=WorkingOU,DC=pucit,DC=local' -JobType
SmartGroup -TriggerType Monthly -StartTime '16:56' -
YearMonths 'March','August','September' -MonthDate 7
```

To use secure credentials, first create them and save them to a variable named 'creds'.

```
$creds = Get-Credential
```

Example 3

This example creates a GUS job by providing a messaging system.

```
New-Schedule -ScheduleName GusTest1 -Targets
'OU=ArslanAhmadOU,OU=WorkingOU,DC=pucit,DC=local' -JobType
GUS -Credentials $creds -TriggerType Daily -StartTime
'16:56' -MessagingSystems 'ARSLANAHMADSVM.PUCIT.LOCAL'
```

Example 4

This example creates a GUS job specifying that it should include all containers and messaging systems.

```
New-Schedule -ScheduleName GusTest2 -IncludeAllContainers -
IncludeAllMessagingSystems -JobType GUS -Credentials $creds
-TriggerType Daily -StartTime '16:56'
```

Example 5

This example creates a job by configuring the notification settings. This commandlet specifies that the notifications for this schedule are enabled and sent to the specified recipients as well as to the owners of the schedule targets. The notifications are only sent when the schedule completes its job successfully.

```
New-Schedule -ScheduleName GusTest3 -IncludeAllContainers -
IncludeAllMessagingSystems -JobType GUS -Credentials $creds
-TriggerType Daily -StartTime '16:56' -EnableNotifications
-Recepients 'recep1@gid.com','recep2@gid.com' -SendToOwners
-NotificationSendingCriteria OnSuccess
```

Remove-Schedule

The commandlet **Remove-Schedule** removes a schedule (by its name or ID) from the identity store connected to the current instance of the Management Shell.

Syntax

```
Remove-Schedule
  -ScheduleName <String>
  [-PassThru]
  [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-WarningAction <ActionPreference>]
  [-InformationAction <ActionPreference>]
  [-WarningVariable <String>]
  [-InformationVariable <String>]
  [-PipelineVariable <String>]
  [<CommonParameters>]
```

```
Remove-Schedule
  -ScheduleId <Int32>
  [-PassThru]
  [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-WarningAction <ActionPreference>]
  [-InformationAction <ActionPreference>]
  [-WarningVariable <String>]
  [-InformationVariable <String>]
  [-PipelineVariable <String>]
  [<CommonParameters>]
```

Required parameters

- ScheduleName or Scheduled

Example 1

This example removes a schedule named GUS811_1.

```
Remove-Schedule -ScheduleName GUS811_1
```


Example 2

This example removes two schedules – GUS1 and GUS2 using the pipeline operator.

```
'GUS_1', 'GUS_2' | Remove-Schedule
```

Example 3

This example removes all schedules with job type Glm.

```
Get-Schedule -JobType Glm | Select-Object -Property Name |
Remove-Schedule
```

Set-Schedule

The commandlet **Set-Schedule** modifies the attributes and settings of a schedule in the identity store connected to the current instance of the Management Shell.

Syntax

```
Set-Schedule
  -ScheduleName <string>
  [-NewName <string>]
  [-TargetOperation {Add | Remove}] [-Targets <string[]>]
  [-Credential <pscredential>]
  [-UserName <string>]
  [-Password <string>]
  [-SetNotifications <bool>]
  [-Recepients <string[]>]
  [-SendToOwners <bool>]
  [-NotificationSendingCriteria {Always | OnSuccess |
OnFailure | OnMembershipChanged}]
  [-Enabled <bool>]
  [-TriggerOperation {add | remove single by id | remove by
type | remove all}]
  [-TriggerId <int>]
  [-TriggerType {Event | Time | Daily | Weekly | Monthly |
MonthlyDOW | Idle | Registration | Boot | Logon |
SessionStateChange | Custom}]
  [-StartTime <datetime>]
  [-MonthDate <int>]
  [-YearMonths {January | February | March | April | May |
June | July | August | September | October | November |
December | AllMonths}]
  [-MonthWeek {FirstWeek | SecondWeek | ThirdWeek |
FourthWeek | LastWeek | AllWeeks}]
  [-WeekDays {Sunday | Monday | Tuesday | Wednesday |
Thursday | Friday | Saturday | AllDays}]
  [-DaysInterval <int>]
```



```
  [-WeeksInterval <int>]
  [-Repeat]
  [-RepeatInterval <int>]
  [-RepeatDuration <int>]
  [-EndDate <datetime>]
  [-TriggerDisabled]
  [-KillAtDurationEnd]
  [-IncludeAllContainers]
  [-IncludeSpecifiedContainers]
  [-MessagingSystems <string[]>]
  [-IncludeAllMessagingServers]
  [-IncludeSpecifiedMessagingServers]
  [<CommonParameters>]
```

Required parameters

- ScheduleName

Example 1

This example renames a schedule from *GUS1* to *GUS1-renamed*.

```
Set-Schedule -ScheduleName GUS1 -NewName GUS1_renamed
```

Example 2

This example updates the authentication information of *GUS1* schedule.

```
Set-Schedule -SscheduleName GUS1 -Credential $creds
```

Example 3

This example removes OU targets from *smm4* schedule.

```
Set-Schedule -ScheduleName smm4_ -TargetOperation Remove -
Targets 'OU=ArslanAhmadOU,OU=WorkingOU,DC=pucit,DC=local',
'OU=CustomRole,OU=WorkingOU,DC=pucit,DC=local',
'OU=CustomRole2,OU=WorkingOU,DC=pucit,DC=local'
```

Example 4

This example modifies smm4_ schedule by removing its targets.

```
Set-Schedule -ScheduleName smm4_ -TargetOperation Remove -
Targets
'CN=STest1Group,OU=ArslanAhmadOU,OU=WorkingOU,DC=pucit,DC=l
ocal', 'OU=CustomRole2,OU=WorkingOU,DC=pucit,DC=local'
```

Example 5

This example clears configured notification settings of a schedule *smm4*.

```
Set-Schedule -ScheduleName smm4_ -SetNotifications $false
```

Example 6

This example changes notification settings of a schedule *smm4*. It sets notification to be sent to *recep1@gid.com* every time the job is run.

```
Set-Schedule -ScheduleName smm4_ -SetNotifications $true -
Recepients 'recep1@gid.com' -NotificationSendingCriteria
Always
```

Example 7

This example adds a monthly trigger for *smm4* schedule. It is repeated every 10 minutes for 1 hour on 23rd of March, August and September at 16:56.

```
Set-Schedule -ScheduleName smm4_ -TriggerOperation Add -
TriggerType Monthly -StartTime '16:56' -MonthDate 23 -
YearMonths 'March,August,September' -Repeat -RepeatInterval
10 -RepeatDuration 60
```

Example 8

This example adds a monthly repeating trigger for *smm4_* schedule and has an end date. It stops if it runs at the duration end.

```
Set-Schedule -ScheduleName smm4_ -TriggerOperation Add -
TriggerType Monthly -StartTime '16:56' -MonthDate 23 -
YearMonths 'March,August,September' -Repeat -RepeatInterval
10 -RepeatDuration 60 -EndDate '2020/03/29' –
KillAtDurationEnd
```

Stop-Schedule

The commandlet **Stop-Schedule** stops a specified schedule if it is already running.

Syntax

```
Stop-Schedule
  [-ScheduleName <String>]
  [-JobId <Int32>]
  [-PassThru]
  [-IdentityStoreId <Int32>]
  [-SecurityToken <CustomClaimsPrincipal>]
  [-WarningAction <ActionPreference>]
  [-InformationAction <ActionPreference>]
  [-WarningVariable <String>]
  [-InformationVariable <String>]
  [-PipelineVariable <String>]
  [<CommonParameters>]
```


Required parameters

- ScheduleName

Example 1

This example stops a schedule *smm4* by name.

```
Stop-Schedule -ScheduleName smm4_
```

Example 2

This example stops a schedule with job type as *GUS*.

```
Stop-Schedule -JobType GUS | Select-Object -Property Name |
Invoke-Schedule
```

Example 3

This example stops all the daily running GUS jobs.

```
Get-Schedule -JobType GUS -TriggerType RunDaily -
MatchingCriteria And | Select-Object -Property Name | Stop-
Schedule
```


Chapter 15 - GroupID Commandlets Parameters

This chapter provides description of parameters of GroupID Management Shell Commandlets covered in this guide.

List of Parameters

The following table lists the GroupID Management Shell commandlet parameters in alphabetical order. Click on alphabet letter to easily locate the parameter which starts with that letter.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Parameter Name	Description
A	
AcceptMessagesOnlyFrom	The distinguished names (DN), globally unique identifiers (GUID) or samAccountNames of the mailbox users and mail-enabled contacts who can send e-mail messages to the group. Providing a blank value enables the group to accept messages from all mailbox users and all mail-enabled contacts. (Applies to Distribution groups only).
AcceptMessagesOnlyFromGroups	The distinguished name (DN), globally unique identifier (GUID) or samAccountName of one or more groups or users that the group is allowed to accept messages from. Separate multiple objects with commas (,). (Applies to Distribution groups only.)
AccidentalDeletion	If the value is set as True, user will be prompted before container deletion.
Add	**Set-User**, **Set-contact**, **Set-Mailbox** Add will append the values of multi-value attributes and replace the value of single-value attributes.
	Set-Group This setting applies to the AdditionalOwners parameter and lets you add one or more additional owners for this group.

<table>
<tr><th>Parameter Name</th><th>Description</th></tr>
<tr><td></td><td>The syntax in which the value is entered for this setting is:
-Add @{ AdditionalOwners = "Owner1","Owner2","Owner3"}
-Add @{ AcceptMessagesOnlyFrom = "User1","User2","User3"}
-Add @{ AcceptMessagesOnlyFromGroups = "Group1","Group2","Group3"}
-Add @{ RejectMessagesFrom = "User1","User2","User3"}
-Add @{ AcceptMessagesOnlyFrom = "Group1","Group2","Group3"}
As the value of objects to be added, the setting accepts all the identities supported by the AdditionalOwners parameter, which is the distinguished name (DN), globally unique identifier (GUID) or samAccountName of the user, contact, or security group.</td></tr>
<tr><td></td><td>Set-SmartGroup, Convert-Group, Set-Dynasty
This setting applies to the following multi-valued parameters and lets you add one or more values to these parameters. Parameters and the syntax for their values follows:
<table>
<tr><th>Parameters</th><th>Syntax</th></tr>
<tr><td>SearchContainers
(StartPaths can be used as an alternative name of this parameter for this setting)</td><td>-Add @{ SearchContainers = "Container1#1","Container2#2"}
Or
-Add @{ StartPaths = "Container1#1","Container2#2"}
Here # specifies the search scope for each container. If scope is not given, then subtree (2) is used as default.</td></tr>
<tr><td>IncludeRecipients
(Includes can be used as an alternative name of this parameter for this setting)</td><td>-Add @{ IncludeRecipients = "Object1","Object2"}
Or</td></tr>
</table></td></tr>
</table>

Parameter Name	Description	
		-Add @{ Includes = "Object1","Object2"}
	ExcludeRecipients (Excludes can be used as an alternative name of this parameter for this setting)	-Add @{ ExcludeRecipients = "Object3","Object4"} Or -Add @{ Excludes = "Object3","Object4"}
	AdditionalOwners	-Add @{ AdditionalOwners = "Owner1","Owner2","Owner3"}
Only Set-Dynasty has this attribute.	**GroupBy**	-Add @{GroupBy= "Attribute1#Container#Filter#Separator", "Attribute2#Container#Filter#Separator"}
	AcceptMessagesOnlyFrom (AuthOrig can be used as an alternative name of this parameter for this setting)	-Add @{ AcceptMessagesOnlyFrom = "User1","User2","User3"} Or -Add @{ AuthOrig = "User1","User2","User3"}
	AcceptMessagesOnlyFromGroups (DLMemSubmitPerms can be used as an alternative name of this parameter for this setting)	-Add @{ AcceptMessagesOnlyFromGroup = "Group1","Group2","Group3"} Or -Add @{ DLMemSubmitPerms = "Group1","Group2","Group3"}
	RejectMessagesFrom (UnauthOrig can be used as an alternative name of this parameter for this setting)	-Add @{ RejectMessagesFrom = "User1","User2","User3"}

Parameter Name	Description	
		Or -Add @{ UnauthOrig = "User1","User2","User3"}
	RejectMessagesFromGroup (DLMemRejectPerms can be used as an alternative name of this parameter for this setting)	-Add @{ RejectMessagesFromGroup = "Group1","Group2","Group3"} Or -Remove @{ DLMemRejectPerms = "Group1","Group2","Group3"}
	The setting accepts all the identities supported by the parameter as the value of objects for each parameter. For example, for the SearchContainer parameter, the setting can accept the DN and GUID of the domains or containers being searched for group members.	
AdditionalOwners	The distinguished name (DN), globally unique identifier (GUID), or samAccountName of one or more users, contacts, or groups (security groups only) to set as the additional owners for the group. Passing a blank value for this parameter will remove additional owners.	
Address	Home address of a user, contact or mailbox.	
AdministrativeNotes	Any information about the group that is useful for its maintenance or administration. It appears on the Exchange Advanced tab of Group Properties dialog box.	
AdminUserName	The admin username for the Google based providers and messaging systems. This parameter becomes available depending on the value of other parameters - *IdentityStoreType* and *Provider*.	
Alias	Alias of user, group or mailbox. The alias parameter can be a combination of characters separated by a period without any spaces. Avoid using special characters in the alias. The Exchange alias is limited to 64 characters, must be unique and should not contain spaces.	

Parameter Name	Description
AliasTemplate	Specifies the pattern for creating alias names for Dynasty children. For a Managerial Dynasty, the template must contain the %MANAGER% keyword in the input string. This keyword is replaced with the respective manager. For all other Dynasties, the value must contain the %GROUPBY% keyword in the input string for replacement with the respective GroupBy value.
All	Perform action on all types of entities.
AppId	Used to provide Azure application ID for Azure / Office 365 based identity stores and messaging systems. Note that this parameter appears depending on the values of other parameters. Application ID which is generated by Azure AD when the application is registered in Azure AD. This parameter becomes available depending on the value of other parameters - *IdentityStoreType* and *Provider*.
Assistant	It will be a DN or (GUID) of another user or contact.
AttributesToLoad	Provide list of attributes which should be loaded with objects. In the absence of the list, object will be loaded with minimal attributes.
AuthenticationMode	Following are the possible values for this parameter: • 1 (credentials of the logged-in users) • 2 (works in conjunction with IdentityStoreID and Credentials parameters). • 3 (user is authenticated through the Log in dialog box which is also the default mechanism if no authentication mode is defined by user).
AuthenticationType	Supported authentication types in GroupID which are: • Security questions • Email • SMS • Yubikey • Windows Hello • Authenticator • Link account • PhoneID

Parameter Name	Description
AuthenticationTypeOperation	Enables or disables the specified authentication type(s).
B	
Business	First business phone number of a user, contact or mailbox.
Business2	Second business phone number of a user, contact or mailbox.
BypassOwnersPolicy	This parameter bypasses the values set in GroupID configurations both for primary owner and required minimum additional owners at group creation or modification. If the value is 0 (zero) then this parameter has no affect.
C	
CarbonCopy	Email address for carbon copy (CC) of notification to be sent other than the main email addresses.
ChangeTrackerActions	The list of GroupID actions to track for history records. The possible values are: • None • AdditionalOwnerChange • Enrollment • ExpirationPolicyChange • GroupExpire_Renew • OobChange • SecurityTypeChange • WorkflowApprovalDenial • OwnershipChange • QueryChange • AllOthers • All • UpgradeSmartGroupChange To track multiple actions, separate each action with a hash (#) sign and set the complete string as a value of this setting. For example, to track changes in additional owners, enrollment details and security types, specify the value as "AdditionalOwnerChange#Enrollment#SecurityTypeChange".
ChildContainer	The distinguished name (DN) or globally unique identifier (GUID) of the container where you want to create the child groups. If you have selected multiple group-by attributes, you can specify a different child container for every attribute in the

<table>
<tr><th>Parameter Name</th><th>Description</th></tr>
<tr><td></td><td>same sequence as the group-by attributes are specified, separating each with a comma (,). For Managerial Dynasty, passing a blank value creates child groups in the container where the top manager resides.</td></tr>
<tr><td>City</td><td>The city of a user, contact or mailbox.</td></tr>
<tr><td>Clear</td><td>Set-User, Set-Contact, Set-Mailbox
It will clear the values of multi-value and single-value attributes.</td></tr>
<tr><td></td><td>Set-Group
This setting applies to the AdditionalOwners parameter and lets you clear the additional owners list.
The syntax for entering the value for this setting is:
-Clear @{ AdditionalOwners}
-Clear @{ AcceptMessagesOnlyFrom }
-Clear @{ AcceptMessagesOnlyFromGroups }
-Clear @{ RejectMessagesFrom }
-Clear @{ AcceptMessagesOnlyFrom }</td></tr>
<tr><td></td><td>Set-SmartGroup, Convert-Group, Set-Dynasty
This setting works for the following multi-valued parameters and lets you clear all their existing values. Parameters and the syntax for their values follows:
<table>
<tr><th>Parameters</th><th>Syntax</th></tr>
<tr><td>SearchContainers
(StartPaths can be used as an alternative name of this parameter for this setting)</td><td>-Clear @{ SearchContainers}
Or
-Clear @{ StartPaths}</td></tr>
<tr><td>IncludeRecipients
(Includes can be used as an alternative name of this parameter for this setting)</td><td>-Clear @{ IncludeRecipients}
Or
-Clear @{ Includes}</td></tr>
<tr><td>ExcludeRecipients
(Excludes can be used as an alternative name of this parameter for this setting)</td><td>-Clear @{ ExcludeRecipients}
Or
-Clear @{ Excludes}</td></tr>
<tr><td>AdditionalOwners</td><td>-Clear @{ AdditionalOwners}</td></tr>
</table></td></tr>
</table>

Parameter Name	Description	
Only Set-Dynasty has this attribute.	**GroupBy**	-Clear @{ GroupBy}
	AcceptMessagesOnlyFrom (AuthOrig can be used as an alternative name of this parameter for this setting)	-Clear @{ AcceptMessagesOnlyFrom } Or -Clear @{ AuthOrig }
	AcceptMessagesOnlyFromGroups (DLMemSubmitPerms can be used as an alternative name of this parameter for this setting)	-Clear @{ AcceptMessagesOnlyFromGroups } Or -Clear @{ DLMemSubmitPerms }
	RejectMessagesFrom (UnauthOrig can be used as an alternative name of this parameter for this setting)	-Clear @{ RejectMessagesFrom } Or -Clear @{ UnauthOrig }
	RejectMessagesFromGroup (DLMemRejectPerms can be used as an alternative name of this parameter for this setting)	-Clear @{ RejectMessagesFromGroup } Or -Clear @{ DLMemRejectPerms }
	As the value of objects for each parameter, the setting accepts all of the identities supported by the parameter. For example, for SearchContainer parameter, the setting can accept the distinguished name (DN) and globally unique identifier (GUID) of the domains or containers to be searched for the group members.	
ClearSet	Clears the specified notification recipients set. Possible values are: • All • Recipients • PasswordExpiry (Password Expiry group notifications) • ML (Membership life cycle notifications) • MB (Managed by life cycle notifications)	

Parameter Name	Description
ClientName	Name of GroupID client such as Automate, Management Shell, GroupID Mobile Service, each Self-Service portal, each Password Center portal.
Company	The company of user, contact or mailbox.
ConfiguredExchange	Specifies the messaging system that GroupID uses for creating the e-mail addresses of mail-enabled objects. The default value 1 uses the latest version of Exchange installed if GroupID is connected to a domain with multiple versions of Exchange. You can change the system to any of the following values: • 2007 (for Exchange 2007) • 2010 (for Exchange 2010) • 2013 (for Exchange 2013) • 2016 (for Exchange 2016) • 2019 (for Exchange 2019) • 0 (for AD-only domain) • 2 (other messaging system)
Connected	Used to request connected identity store to the current instance of GroupID Management Shell.
Container	The distinguished name (DN) or globally unique identifier (GUID) of one or more containers where you want to search for a user, contact or group. Separate multiple values with commas.
Country	Country of a user, contact or mailbox, represented as the 2-character country code based on ISO-3166.
CreateFlatManagerialList	Setting a True value creates this dynasty as flat managerial list. A flat managerial list is a form of managerial dynasty in which all direct reports of the top manager and sub-level managers are added as members of one group and no separate groups are created for the sub-ordinates of the top manager's direct reports. If this setting is set to True, the flat operation is performed on the next update of the dynasty where it breaks its current hierarchy and re-builds the memberships of the parent group on the flat dynasty logic. (Applies to Managerial Dynasty)
CriteriaFilters	Same as RoleCriteriaFilters
CriteriaScope	Same as RoleCriteriaScope

Parameter Name	Description
Credential	The $Credentials environment variable holds the user's authentication information. Use this variable to execute the commandlet using the credentials of a user account other than the one you are logged on to the connected identity store.
CustomAttribute1-15	A value for an attribute that you determine. Use these attributes—up to 15—to store additional information specific to your needs.
D	
Database	SQL database name of previous GroupID version.
DataSourceConnection	Set or modify connection string of an external data source in Query Designer of a Smart Group or Dynasty.
DataSourceName	The name of the database that contains the table or view you want to use for your query. This parameter is applicable on the following data source types: • Microsoft SQL Driver • Oracle
DataSourcePassword	The password for the specified user account to use for connecting to the specified data source.
DataSourceQuery	Specifies the database query to execute to retrieve results from the data source. This can be a query statement and can include multiple columns separated by commas (,). The field names are enclosed in brackets ([ ]) to prevent any ambiguity that the query engine might encounter because of spaces between column names. GroupID Management Shell also needs to know how the information in the source relates to the directory so it can find the recipients identified in the data source in the directory and add them to the group. This relation is defined through the LdapFilter parameter. If no match is found, the data source entry will be skipped.
DataSourceType	Use this parameter to combine an external data source with Active Directory to determine the group membership. When a connection is configured, GroupID Management Shell connects to the database and retrieves results. It then queries Active Directory to find matching records. The parameter can also be used to connect to

Parameter Name	Description
	external directories. Specify any of the following external data source types: • Text Driver • ODBC Data Source • Sun ONE iPlanet Driver • Lotus Notes • Microsoft SQL Driver • Oracle
DataSourceUserName	The username of the account to use for connecting to the specified data source.
DaysInterval	Specified the daily interval for daily triggers.
DefaultAllowPermissions	By default, all permissions except those specified in *RolePermissions* are denied. The application of this parameter overrides the default behavior and causes all of the permissions except those specified in *RolePermissionNames* to be granted.
DefaultApprover	Specifies the default approver for an identity store.
DefaultExpirationPolicy	The default expiry days to set for new groups at creation, which can later be changed for groups individually using the Set-SmartGroup commandlet. The default value 0 implies that the groups will never expire.
DefaultGroupApprover	The distinguished name (DN), globally unique identifier (GUID) or samAccountName of the default approver to whom notifications will be sent for groups having no owners.
DefaultGroupDeletionTimeAfterExpiry	The number of days after which an expired group should be deleted. The default value is 30. This parameter only applied if the value of the DeleteExpiredGroups parameter has been set to True.
DefaultMaximumNumberOfMembers	The maximum number of members a group can have.
DefaultMaximumNumberOfMembersToDisplay	The maximum number of items to display in the Automate groups list. The default limit is set to 1000.
DefaultNumberOfOwnersToDisplay	The number of most recently used recipients (set as group owners) to show on the shortcut menu when setting the owner for multiple groups. The default value is 5.

Parameter Name	Description
DefaultReportToMessageOriginator	Setting its value to True sends non-delivery reports (NDR) to the message originator (sender). By default, it is set to False.
DefaultReportToOwner	Setting its value to True sends non-delivery reports (NDR) to the group owner. By default, the value is set to False.
DefaultRequestDeletionTime	Workflow requests older than the number of days given in this parameter will be deleted by the CleanupApprovedRequests, CleanupDeniedRequests and CleanupPendingRequests settings. The default value of this setting is 30. **Note:** This setting applies only if the DeleteRequests setting is set to True.
DefaultStartWithGlobalCatalogInQueryDesigner	Its default value True sets the Global Catalog as the default scope for searches on the Query Designer. Changing its value to False searches the logged-on domain only.
DefaultUnusedGroupsExpirationTime	This setting is related to the group usage lifecycle and applies only if the GroupUsageLifecycleEnabled and ExpireUnusedGroups settings are set to True. Its value is the unused period (in number of days) of the lifecycle period for a mail-enabled distribution group after which its life is reduced to 7 days. The default value of this setting is 60 days.
DeletedObjects	It is a switch, if present then delete object replication will be started.
DeleteEmpty	Setting its value to True forces Automate to delete Dynasty children when they are empty or when their parents are deleted. The default value is False.
DeleteExpiredGroups	The default value True enables the automatic deletion of expired groups according to the number of days specified in the DefaultDeletionTimeAfterExpiry parameter.
DeleteNestedOrphanGroups	This parameter deletes nested orphan groups according to the following rules: • If the maximum membership value is reached and the Do not update option is selected, then the parameter has no effect.

Parameter Name	Description
	• If the maximum membership value is reached and the Nest into child groups option is selected, then, upon membership update, more nested child groups are created and orphan nested groups are deleted. • If the maximum membership value is increased then upon the group's membership update, members from the nested child groups are moved into the parent group and the nested groups are orphaned. This parameter deletes the nested groups.
DeleteRequests	The default value True enables the removal of older workflow requests, a feature that removes those approved, pending, and denied workflow requests that are older than the number of days specified in the DefaultRequestDeletionTime setting.
Department	The department of a user, contact or mailbox.
Description	Used to provide description of an entity while: • creating a new group (managed or unmanaged) or dynasty. • modifying a user, contact, group (managed or unmanaged) or dynasty. • converting a static group to a smart group.
DestinationContainer	The distinguished name (DN) or globally unique identifier (GUID) of the container that you want to move the group to. The destination container must be part of the same forest.
DirectReports	Provide any of the following identity for the direct report: • Distinguished name (DN) • Globally unique identifier (GUID) • Comman-name (Cn) • Name • SamAccountName
DisableAttributeUpdation	Specifies that attribute updation should not occur when Profile Validation cycle of a user is expired.
Disabled	In some commandlet this parameter is used to retrieve disabled entities such as disabled schedules or identity stores and in some it disables an entity.
DisableExpiredGroupDeletion	Disables the deletion of the expired groups.

Parameter Name	Description
DisableGroupAttestation	Disables the group attestation at identity store level.
DisableGUSLifecycle	Disables the group usage life cycle of groups at identity store level.
DisableNewProfileValidationLifecycle	Disables the profile validation of new profiles.
DisableOrphanGroupDeletion	Disables deletion of orphan groups when, upon membership update, they become orphan.
DisableOutOfBoundsAlerts	Disables generation of out of bound alerts to group owners upon membership threshold and does not update the membership.
DisableSecurityGroupsExpiry	Disables expiration of the security groups.
DisableSWAuthenticationViaEmail	Disables second way authentication via email.
DisableSWAuthenticationViaMobile	Disables second way authentication via mobile.
DisableSWAuthenticationViaSecurityQuestions	Disables second way authentication via security questions.
DisableValidationDateRemoval	Causes the validation date not to be cleared after the profile validation has been expired.
DisallowingPasswordExceptionFilePath	Specifies the path to a file containing a list of strings that cannot be set as password.
DisplayName	Display name while • creating a user, contact, group (managed & unmanaged), dynasty or mailbox. • modifying a user, contact, group (managed & unmanaged), dynasty or mailbox. • converting a static group to a smart group. • retrieving a tombstone object.
DisplayNameTemplate	Specifies the pattern for generating display names for Dynasty children. For the Managerial Dynasty, the template must contain the %MANAGER% keyword in the input string. This keyword is replaced with the respective manager. For all other Dynasties, the value must contain the %GROUPBY% keyword in input string for replacement with the respective GroupBy value.
DistinguishedName	Distinguished name of an object in directory.
Domain	Domain name of the provider mentioned in a commandlet. The domain name can be of an Active Directory domain, Azure domain or messaging provider's domain. This parameter

Parameter Name	Description
	becomes available depending on the value of other parameters.
DomainExpiration	(Applies to Password Expiry group.) The domain expiration policy for the group. This policy allows you to specify maximum password age. The default value is 42 days.
DynastyManagerAsMember	Set its value to True to add the manager of direct reports to the membership of the direct reports group so that the manager receives a copy of any e-mail sent to the group. The default value is False.
E	
EmailAddress	A valid email address of a user, contact, mailbox or group (if mail-enabled)
EmailProviderDomain	This setting applies if the ConfiguredExchange setting is set to 2. Its value is the domain name of the external e-mail provider. For example, googlegroups.com.
EmailTemplatePath	Location of the email template that will be used while sending an email notification to a user or group.
EnableAttributeUpdation	Enables attribute update when a user is expired in Profile Validation cycle. It sets the given string as the attribute's value for the user.
Enabled	In some commandlet this parameter is used to retrieve enabled entities such as enabled schedules or identity stores and in some it enables an entity.
EnableExpiredGroupsDeletion	Enables the deletion of expired groups.
EnableGroupAttestation	Enables the group attestation i.e. to review and validate the attributes and membership of an expiring group before renewing it.
EnableGUSLiefecycle	Enable group usage life cycle i.e. set the expiry of mail-enabled distribution groups based on their usage.
EnableNewProfileValidationLifecycle	Enables profile validation for newly found user objects (by way of newly created objects or by way of disabled object enabled again) in the directory.
EnableNotifications	Enables notifications in a schedule.

Parameter Name	Description
EnableOrphanGroupDeletion	Enables deletion of orphan groups when, upon membership update, they become orphan.
EnableOutOfBoundsAlerts	Enables generation of out of bound alerts to group owners upon membership threshold and does not update the membership.
EnableSecurityGroupsExpiry	Enables expiry of security groups.
EnableSWAuthenticationViaEmail	Enables second way authentication via email.
EnableSWAuthenticationViaMobile	Enables second way authentication via mobile.
EnableSWAuthenticationViaSecurityQuestions	Enables second way authentication via security questions.
EnableUpdate	Specify False to disable the group update and scheduled job process. Default value is True.
EnableValidationDateRemoval	Clears the validation date if X number of days have passed since the last validation date. In case of a rehire scenario, the object will be treated as a newly created object and the validation process for new users will apply to it.
EndDate	Date on which membership will end or restore. Or Date on which membership will end/restore, or a schedule will end.
EnforceOutOfBounds	Set its value to True to break a group into nested child groups when it reaches the maximum membership limit, specified in the *DefaultMaximumNumberOfMembers* parameter. The default value False prevents any action from being taken when the membership limit is reached.
EnrollmentEnabled	Enables / Disables enrollment on an identity store.
EnrollmentType	Possible values are: • None • Mobile • SecurityQuestions • Email • Authenticator • LinkAccount • PhoneID • Yubikey • WindowsHello • All

Parameter Name	Description
	• Any
ExcludeNestedLists	Setting a True value excludes child Dynasties from the membership of the parent Dynasty. The default structure of Managerial Dynasty adds the Smart Group of sub-level manager in the membership list of the top-level manager's Smart Groups. (Applies to Managerial Dynasty)
ExcludeOUs	The default value True excludes from exploration the organizational units specified in the IncludeExcludeOUs parameter. Setting its value to False applies the expiration only on the organizational units specified in the IncludeExcludeOUs parameter and excludes the rest.
ExcludeRecipients	The distinguished name (DN), globally unique identifier (GUID) or samAccountName of one or more objects that you want to exclude statically from the group membership regardless of whether they are returned by the query.
ExpansionServer	The name of the Expansion server. The Expansion server is the Exchange server responsible for expanding a distribution list and creating a message for each of the members.
ExpirationPolicy	Set the expiration policy for the group. This parameter does not work for Dynasty children since they inherit the expiration policy of their parent Dynasty and you cannot change it explicitly at child level.
ExpirationRange	The expiration range policy for the group. This policy defines when GroupID Management Shell will include a user in the membership of the Password Expiry group. For example, a domain expiration policy is configured with a maximum password age of 30 days. Setting the expiration range policy to 10 will include users in the membership of the Password Expiry group who have passwords aged 20 days or older. (Applies to Password Expiry group)
ExpiredGroupsDeletionInterval	Number of days since groups expiry after which the groups shall be deleted.
ExpireUnusedGroups	This setting is related to the group usage lifecycle and applies only if GroupUsageLifecycleEnabled

Parameter Name	Description
	is set to True. The value True reduces the life of mail-enabled distribution groups that have not been sent any e-mail for a particular period. This unused period is defined in the DefaultUnusedGroupsExpirationTime setting. Under its default value False, the life of unused groups is always extended as soon as they reach their expiration date.
ExtendGroupLife	Extend the life of the group as per the ExpirationPolicy parameter's value. The default value of this parameter is True, so specifying a value is not required.
ExtensionDataAttributes	By default, ExtensionDataAttribute attribute is used for storing the value. In case it has been modified then this parameter must specify the attribute being used for storing the value.
F	
FileLoggingEvent	Set the event for which file logs are generated.
FilePath	The path of the text file, if the value of the DataSourceType parameter is Microsoft Text Driver.
FilterOperation	Operation to perform on role criteria filters
Filters	Specifies how the values of group-by attributes are stripped out for creating the child groups. This parameter allows you to collapse several different values into one. Use any of the following as a value of this parameter: • <Blank value> - Do not use any filter and create a group for each distinct value of the attribute. • Left <Number of characters> - Selects the specified number of characters from the attribute starting from the left-end of the string. Each distinct set of selected characters from the group-by attribute is then used to create a group. • Right <Number of characters> - Selects the specified number of characters from the attribute starting from the right-end of the string. Each distinct set of selected characters from the group-by attribute is then used to create a group.

Parameter Name	Description
	• %GROUPBY%/<the part of the value to leave out> - Use this filter when you have a character separator. Specifying this filter creates a group for each distinct value of the portion of the attribute selected. %GROUPBY% represents the significant portion of the value. After the slash, you can specify the portion you want to leave out of the attribute's value. Specifying * after the slash leaves out any portion of the value that occurs after the slash. For multiple group-by attributes, provide a filter values for each attribute separated by a comma (,).
FirstName	The first name of a user, contact or mailbox.
FromEmail	Email address that SMTP uses to send emails from.
FromEmailAddress	The e-mail address to use for sending notifications
G	
GenerateOnedayToExpiryReport	The default value True notifies the group owner of its expiry one day before the expiration date. Set its value to False to disable this notification.
GenerateSevenDaysToExpiryReport	The default value True notifies the group owner of its expiry seven days before the expiration date. Set its value to False to disable this notification.
GenerateThirtyDaysToExpiryReport	The default value True notifies the group owner of its expiry thirty days before the expiration date. Set its value to False to disable this notification.
GroupAlias	Alias for the new group, distribution group or dynasty. The alias can be a combination of characters separated by a period without any spaces. Avoid using special characters in the alias. The Exchange alias is limited to 64 characters, must be unique and should not contain spaces.
GroupBy	Name of the group-by attribute. Separate multiple attributes with commas (,). This parameter is required for all Dynasties except the Managerial Dynasty.
GroupIdentity	The distinguished name (DN), globally unique identifier (GUID), security identifier (SID),

Parameter Name	Description
	canonical name (CN) or SamAccountName of the group to add members to.
GroupIDVersion	Previous GroupID version to upgrade from. This parameter accepts integer values e.g. 7.0, 8.0 and 9.0. • 7.0 = GroupID 7.0 • 8.0 = GroupID 8.0 • 9.0 = GroupID 9.0
GroupLifeDays	Specifies the number of days to extend / reduce (depending on the configured extension policy) if the group has not been used this number of days.
GroupNamePrefixes	One or more prefixes configured in GroupID configurations. They are prefixed with the group name and display name when you create a new group or modify an old group using the Properties option.
GroupScope	Specify the scope for the group or dynasty. The available group scopes are: Universal, Global, and Domain Local.
GroupType	Specify the group types for upgrade: 1 = Non-managed groups 2 = Smart Groups 3 = Parent Dynasty 4 = Middle Dynasty 5 = Leaf Dynasty 6 = Password Expiry Smart Group **Note:** When a specific dynasty is upgraded it is recommended to upgrade the whole dynasty using the SearchContainer parameter and update it after running the Upgrade-Group command (provided that the whole Dynasty is in the same container). If a specific parent or middle or leaf Dynasty is upgraded using the Upgrade-Group command, update will be required to link it with the Dynasty chain (provided that all the Dynasties are upgraded to GroupID 8.1).
GroupUsageLifecycleEnabled	Set its value to True to enable the group usage lifecycle feature. This lifecycle is executed by Group Management Service (GMS) for mail-enabled distribution groups and adds an additional rule to their regular expiration process. Under this lifecycle, if no e-mail is sent to a mail-

Parameter Name	Description
	enabled distribution group for a particular period, you can set GMS to reduce its expiration date to 7 days. Under its default behavior, unused distributions groups are never expired. As soon as, they reach their expiration date, their life is extended by reapplying the expiration policy on them.
H	
HavingNotifications	Used to select those schedules having notifications enabled. Used only in Get-Schedule
HiddenFromAddressListEnabled	Specifying a True value prevents the group from appearing in Exchange address lists. The default value is False.
HideMembership	Setting its value to True hides group membership in the Outlook address book. The default value is False.
HideMembershipFromAddressListEnabled	A True value prevents the group membership from appearing in the Outlook address book. The default value is False.
HistoryActionsOperation	The operation on actions that the history will keep track of. Possible values are: • Add • Remove • Remove all
HistoryRetention	Specifies the interval for which the history is tracked. Possible values are: • All • Last_30_Days • Last_60_Days • Last_90_Days • Last_120_Days • Last_6_Months • Last_1_Year • Last_2_Years • Last_5_Years
HistorySelectedActions	The actions that the history will keep track of. Possible values are: • OwnershipChange • AdditionalOwnerChange • ExpirationPolicyChange • GroupExpireRenew • QueryChange • SecurityTypeChange

Parameter Name	Description
	• ObjectCreated • ObjectDeleted • IdentityStoreHistory • SecurityRolesHistory • WorkflowsHist
HistoryTrackingOption	Specifies what the history will keep track of. Possible values are: • Nothing • All_Actions • Selected_Actions
Home	First home phone number of a user, contact or mailbox.
Home2	Second home phone number of a user, contact or mailbox.
HomePage	The link of a user, contact, group or mailbox's profile or home page.
I	
Identity	Supported identities are: • Distinguished name (DN) • Globally unique identifier (GUID) • Comman-name (Cn) • Name • SamAccountName
IdentityStoreId	Unique identifier of identity store.
IdentityStoreName	Name of an identity store.
IdentityStoreType	Specify the type of an identity store. Possible types are: • ActiveDirectory • WindowsAzure • GSuite
IgnoreConnectionFail	While creating an identity store, an active service account and valid credentials are required for connecting to an identity store. This parameter overrides this behavior and creates the identity store even if the connection is not active or the credentials are invalid.
IncludeAllContainers	Applies when *JobType* is set to GUS. This parameter includes all containers in the schedule.
IncludeAllMessagingSystems	Applies when *JobType* is set to GUS. This parameter includes all messaging systems in the schedule.

Parameter Name	Description
IncludeDisabledUsers	(Applies to Password Expiry group.) Specifying this parameter includes disabled users in the group membership.
IncludeEntityTypes	Used only in Get-*RolePermissionNames*. This parameter retrieves the permission categories alongwith the permission name.
IncludeExcludeOUs	The distinguished name (DN) or globally unique identifier (GUID) of one or more organizational units to include in or exclude from expiration. The behavior of this setting depends on the value set for ExcludeOUs parameter.
IncludeManagerAsMember	Setting a True value includes each manager as a member of their direct reports group; so that, whenever an e-mail is sent to the direct reports group, their manager also receives a copy of it. (Applies to Managerial Dynasty only) **Note:** If this setting is set to True, the manager will be included to the membership of direct reports on the next update of the dynasty.
IncludePasswordNeverExpireUsers	Specifying this parameter includes users whose password never expires in the group membership. Skipping this parameter excludes them from the group membership. (Applies to Password Expiry group)
IncludeRecipients	The distinguished name (DN), globally unique identifier (GUID) or samAccountName of one or more objects that you want to include statically in the group membership regardless of whether they are returned by the query.
InheritanceBehavior	Specifies whether Dynasty children should inherit attributes from their parent. The attributes that Dynasty children inherit are stored in the InheritedAttrs option, which can be viewed using the Get-Options commandlet. Values are: • 0 (Inherit selected attributes only on creation) • 1 (Always inherit selected attributes) • 3 (Never inherit selected attributes)
InheritedAttrs	One or more attributes of the parent Dynasty whose values you want its children to inherit at creation or when it is updated.
Initials	The initials of a user, contact or mailbox.
InlineImageFile	The path of the image file that you want to include in the e-mail notification. This image is

Parameter Name	Description
	included in the e-mail body; it is not sent as an attachment.
IsExpired	A True value of the parameter expires the group and a False value renews the group. This parameter does not work for Dynasty children since they expire with the parent.
IsPasswordExpiryGroup	Specifying this parameter creates a Password Expiry group. If skipped, a simple Smart Group will be created.
IsPasswordExpirySmartDL	Specifying this parameter is mandatory if you are updating a Password Expiry group. If this parameter is skipped, the group will be converted to a simple Smart Group.
IsPreciseSearch	If object types parameter is defined, IsPreciseSearch will force search results for those particular object types only.
IsSecurityGroupExpirationPluginEnabled	Set its value to True to enable the security group expiration feature. By default, it is set to False.
J	
JobType	Type of the schedule (e.g. SmartGroup, GUS etc.). This parameter is used in some cmdlets to retrieve the schedules by job type. In *New-Schedule*, it is used to set the type of schedule.
K	
KeepHistoryOption	Specifies the length of time to retain history records in the GroupID database. The default value 0 retains all history data of the actions specified by the ChangeTrackerActions setting. You can change it to any of the following values: • 1 (for 30 days) • 2 (for 60 days) • 3 (for 90 days) • 4 (for 120 days) • 5 (for 6 months) • 6 (for 1 year) • 7 (for 2 years) • 8 (for 5 years) The setting does not destroy the older history data. Rather, it exports the older data to an Excel file for later reference. This Excel file is created in the HistoryBin folder in the GroupID installation directory. Group Management Service performs

Parameter Name	Description
	the history data export. With every execution of the service, it checks the specified period against Keep History option for the domain and exports the older data to the Excel file (if found).
KeepUserHistory	It upgrades the history of the groups.
KeyMapAD	Specify the primary key for provider in external data source in Query Designer.
KeyMapDB	Specify the primary key for Database in external data source in Query Designer.
KillAtDurationEnd	The schedule job will be forced to terminate if it's still running at the end of its duration.
L	
LastName	The last name of user, contact or mailbox.
LdapFilter	The LDAP search filter that defines your search criteria. This parameter stores your query. A Smart Group can dynamically build its membership according to the query associated with it. Similar to Smart Group, a Dynasty has the capability to dynamically build its membership according to the query associated with it.
LDAPSearchContainer	The container for the Sun ONE iPlanet data source.
M	
MailEnabled	Specifies whether to create a mail-enabled user, contact or group (managed & unmanaged). Provide a True value for mail-enabled object, otherwise a non-mail-enabled object will be created.
MailBoxStore	Specifies which mailbox store will be used.
ManagedBy	The distinguished name (DN), globally unique identifier (GUID) or samAccountName of the user, contact or group (security groups only) that you want to set as the group owner or manager. Passing a blank value for this parameter will remove the manager.
Manager	Provide any of the following identity for the manager of the user: • Distinguished name (DN) • Globally unique identifier (GUID) • Comman-name (Cn) • Name

Parameter Name	Description
	• SamAccountName
MatchingCriteria	Used in *Get-Schedule*. Number of criteria (for example *TriggerType* and *JobType*) can be used to retrieve schedules, this parameter describes how to join the criteria by using *Or* and *And*.
MaximumMembersPerGroup	Specifies the maximum number of members that a group can hold. If this limit is reached, out-of-bounds configurations are applied to the group.
MaximumMembersToDisplay	The number of members to display for a group on the Members tab.
MaximumPasswordAge	The parameter has no effect on the group to be modified.
MaxItemsToDisplay	The maximum number of objects the commandlet should return.
MaxiumumAliasLength	This setting works if the ConfiguredExchange setting is set to 2. Its value is the maximum number of characters that an external e-mail alias can contain. The minimum value is 10. The default value is 64.
MaxSendSize	The maximum allowed e-mail message size in kilobytes (KB) that can be sent from the group. (Applies to Distribution groups only)
MembershipCountThreshold	Triggers an out-of-bound exception if the number for current or new membership exceeded than the specified number.
MembershipPercentageThreshold	Specifies that if out-of-bounds alerts are enabled, membership should stop if updation would cause this percentage of members change in the group and generate a notification to owners.
MessagingSystems	Applies when JobType is set to GUS. Use this parameter to specify the message systems for GUS job. This parameter and IncludeAllParameters cannot be applied both at the same time.
MinimumPasswordAge	The parameter has no effect on the group to be modified.
Mobile	Cell number of user, contact or mailbox.
MsExchCoManagedByLink (*ExchangeAdditionalOwners can also be used as an*	The distinguished name (DN), globally unique identifier (GUID) or samAccountName of one or more users that you want to set as Exchange additional owners. This setting applies only if

Parameter Name	Description
alternate name of this parameter)	Exchange Server 2010 is deployed in your environment.
MsExchRequireAuthToSendTo	Set its value to True if you want senders to be authenticated for sending e-mails.
N	
Name	The name of the new organizational unit, group, query-based-distribution group or dynasty being created. **Get-ImanamoCommand** Gets information only about commandlets or command elements with the specified name. Wildcard search is also supported.
NewName	New name of an identity store or a schedule.
NewProfileValidationLifecycle	The number of days within which new users should validate their profiles.
Notes	Description text about a user, contact, group, or mailbox that appears on the General tab of their Properties dialog box.
NotificationSendingCriteria	When a notification for a scheduled job is to be sent. Possible values are: • Always • OnSuccess • OnFailure
NotifyAddedMembers	Notify objects when they are added to the membership of a group.
NotifyLoggedInUsers	Specify whether the logged in users should be notified for changes they make to directory objects using Automate, Self-Service portal, Management Shell, GroupID mobile app, and Password Center portal. This setting applies only to mail-enabled users.
NotifyModifiedObject	Specify whether to send email notification to an object (group, user, contact) being modified. For group, group members are notified. For contact and user, the particular contact or user is notified about the changes.
NotifyOptOutAdditionalOwners	Excludes some or all additional owners from receiving all expiry deletion and renewal notifications.
NotifyOwners	Specifies whether the to send notification emails to the primary and additional owners (for groups),

Parameter Name	Description
	and managers of users/contacts about changes made to the respective objects.
NotifyPublicGroupOwner	Specify whether to send email notifications to the primary and additional owners of a public group upon membership change.
NotifyUserGroupJoinMB	Specify whether to send email notification to users when they are added as additional owner or manager to the membership of a group.
NotifyUserGroupJoinML	Specify whether to send email notification to users when they are added in a group.
NotifyUserGroupLeaveMB	Specify whether to send email notification to users when they are removed as additional owner or manager of a group.
NotifyUserGroupLeaveML	Specify whether to send email notification to users when they are removed as member from a group.
Noun	Shows information about commandlets or command elements having the specified noun in their name. Wildcard search is also supported.
NumberofOwnersToDisplay	The maximum value that can be set for the DefaultNumberOfOwnersToDisplay parameter. 24 is the maximum.
O	
Office	Office phone number of a user, contact or mailbox.
Operator	Same as RoleCriteriaOperator
Options	The list of options to be retrieved from the registry.
OrganizationalUnit	The distinguished name (DN) or globally unique identifier (GUID) of the container where you want to create a user, contact, group or mailbox.
OutOfBoundsAlertEnabled	Set to True to enable out-of-bound exceptions when group memberships change. Out-of-bound exceptions prevent massive changes from occurring to group memberships. When an out-of-bounds exception occurs, the group membership is not updated and the owner or administrator is notified via e-mail. If the owner or administrator determines that the change is valid they can update the group manually.
OutOfBoundsMinimum	This setting works in conjunction with OutOfBoundsPercentage. If both the percentage

Parameter Name	Description
	and the current membership or new membership exceeds the number specified for this parameter, an out-of-bounds exception will occur. The setting applies only if the OutOfBoundsAlertEnabled parameter is set to True.
OutOfBoundsPercentage	The out of bound percentage that is calculated by adding the number of members being added to the group and the number of recipients being removed from the membership and then dividing the result by the total number of new members. This setting works in conjunction with OutOfBoundsMinimum. If both the percentage and the OutOfBoundsMinimum limit is exceeded, an out-of-bounds exception will occur. The setting applies only if the OutOfBoundsAlertEnabled parameter is set to True.
P	
P12CertificatePath	Specify the location of a P12 certificate file for a Google based identity store. Note that this parameter appears depending on the values of other parameters.
PageSize	The number of history records to show on a page on the History tab of the group Properties dialog box.
ParentContainer	The distinguished name (DN), globally unique identifier (GUID) or security identifier (SID) of the container where you want to create a new organizational unit. To create the container at root level, pass the DN of the domain as the value of the parameter.
Password	Password of SQL user name.
PasswordCenterSupportURL	The default URL of the online help for Password Center portals. This URL is set by default for all new portals created using Password Center.
PasswordExceptionOperation	The operation to perform on the values supplied in the *PasswordExceptions* parameter.
PasswordExceptions	Specifiy the password exceptions. This parameter accepts 2-Length arrays. First index contains the operator and the second index contains the value. Possible values for operator are: • Equals

Parameter Name	Description
	• Startswith • Endswith • Contains • Regexp Example: @('contains', 'webdir123R) is a valid value
PasswordPortalUrl	Specify the Password Portal Url.
PasswordRuleOperation	The action to perform on the values supplied in the *PasswordRules* parameter.
PasswordRules	Specify the regular expressions (rules) for passwords.
PermissionOperation	The operation to perform on the *Permissions* parameter.
Permissions	Same as RolePermissions.
Port	Specify the port number for the specified data source.
PowerTools	Include respective power tools to execute script in Query Designer of Smart Group.
ProfileValidationGroupDN	Specify the distinguished name of a group to apply profile validation on.
ProfileValidationReminderOperation	Specify the operation to perform on the value of the *ProfileValidationReminders* parameter.
ProfileValidationReminders	Specify the profile validation reminders. Values are supplied as 2-length array. The first index contains the name of reminder and the second index contains the number of days the reminder is sent to the user relative to the days left for the profile validation period to end. Example: @'first', 15) indicates a reminder named first with 15 days
Provider	Specify a provider for messaging server. The supported providers are: • Office 365 • Suite • Exchange 2010 • Exchange 2013 • Exchange 2016 • Exchange 2019
Q	
QuestionOperation	Specify the operation to perform on the *SecurityQuestions* parameter.

Parameter Name	Description
QueueEmail	Specifying this parameter sends the notification e-mail through Imanami Email Service. Imanami Email Service maintains a queue of all notifications to be sent by GroupID and ensures that they are delivered when the SMTP server is down. If this parameter is left out, the notification e-mail is sent directly without being added to the notification queue. Consequently, if the configured SMTP server is down, the e-mail is lost. Therefore, it is recommended that you use this parameter in every Send-Notification command.
R	
Recipients	Specify recipients for the job completion email notifications.
RegularProfileValidationLife cycle	Specify the number of days for the profile validation life cycle period.
RejectMessagesFrom	The distinguished names (DN), globally unique identifiers (GUID) or samAccountNames of the mailbox users and mail-enabled contacts who are not allowed to send e-mail messages to the group. (Applies to Distribution groups only)
RejectMessagesFromGroup	The distinguished name (DN), globally unique identifier (GUID) or samAccountName of one or more groups or users, the group is restricted to accept messages from. Separate multiple 0bjects with commas (,). (Applies to Distribution groups only.)
Remove	**Set-User**, **Set-Contact**, **Set-Mailbox** It will remove the values of specified attributes.
	Set-Group This setting applies to the AdditionalOwners parameter and lets you remove one or more additional owners for this group. The syntax in which the value is entered for this setting is: -Remove @{ AdditionalOwners = "Owner1","Owner2","Owner3"} -Remove @{ AcceptMessagesOnlyFrom = "User1","User2","User3"} -Remove @{ AcceptMessagesOnlyFromGroups =

<table>
<tr><th>Parameter Name</th><th colspan="2">Description</th></tr>
<tr><td></td><td colspan="2">"Group1","Group2","Group3"}
-Remove @{ RejectMessagesFrom = "User1","User2","User3"}
-Remove @{ AcceptMessagesOnlyFrom = "Group1","Group2","Group3"}
As the value of objects to be removed, the setting accepts all the identities supported by the AdditionalOwners parameter, which is the distinguished name (DN), globally unique identifier (GUID) or samAccountName of the user contact, or security group.</td></tr>
<tr><td></td><td colspan="2">Set-SmartGroup, Convert-Group, Set-Dynasty
This setting applies to the following multi-valued parameters and lets you remove one or more values from these parameters. Parameters and the syntax for their values follows:</td></tr>
<tr><td></td><td>Parameter</td><td>Syntax</td></tr>
<tr><td></td><td>SearchContainers
(StartPaths can be used as an alternative name of this parameter for this setting)</td><td>-Remove @{ SearchContainers = "Container1","Container2"}
Or
-Remove @{ StartPaths = "Container1","Container2"}</td></tr>
<tr><td></td><td>IncludeRecipients
(Includes can be used as an alternative name of this parameter for this setting)</td><td>-Remove @{ IncludeRecipients = "Object1","Object2"}
Or
-Remove @{ Includes = "Object1","Object2"}</td></tr>
<tr><td></td><td>ExcludeRecipients
(Excludes can be used as an alternative name of this parameter for this setting)</td><td>-Remove @{ ExcludeRecipients = "Object3","Object4"}
Or
-Remove @{ Excludes = "Object3","Object4"}</td></tr>
<tr><td></td><td>AdditionalOwners</td><td>-Remove @{ AdditionalOwners = "Owner1","Owner2","Owner3"}</td></tr>
</table>

Parameter Name	Description	
Only Set-Dynasty has this attribute.	**GroupBy**	-Remove @{Groupby=title}
	AcceptMessagesOnlyFrom (AuthOrig can be used as an alternative name of this parameter for this setting)	-Remove @{ AcceptMessagesOnlyFrom = "User1","User2","User3"} Or -Remove @{ AuthOrig = "User1","User2","User3"}
	AcceptMessagesOnlyFromGroups (DLMemSubmitPerms can be used as an alternative name of this parameter for this setting)	-Remove @{ AcceptMessagesOnlyFromGroup = Group1","Group2","Group3"} Or -Remove @{ DLMemSubmitPerms = "Group1","Group2","Group3"}
	RejectMessagesFrom (UnauthOrig can be used as an alternative name of this parameter for this setting)	-Remove @{ RejectMessagesFrom = "User1","User2","User3"} Or -Remove @{ UnauthOrig = "User1","User2","User3"}
	RejectMessagesFromGroup (DLMemRejectPerms can be used as an alternative name of this parameter for this setting)	-Remove @{ RejectMessagesFromGroup = "Group1","Group2","Group3"} Or -Remove @{ DLMemRejectPerms =

Parameter Name	Description
	"Group1","Group2","Group3"}
	The setting accepts all the identities supported by the parameter as the value of objects for each parameter. For example, for the SearchContainer parameter, the setting can accept the DN and GUID of the domains or containers being searched for group members.
Repeat	Repeats the trigger.
RepeatDuration	Applicable only when the Repeat parameter is applied. It specifies the duration in minutes during which the trigger will repeat.
RepeatInterval	Applicable only when the Repeat parameter is applied. It specifies the interval in minutes after which the trigger will start again.
Replace	**Set-User**, **Set-Contact**, **Set-Mailbox** It will replace the old value of attribute with newly specified value.
	Set-Group This setting applies to the AdditionalOwners parameter and lets you entirely overwrite its existing values. The syntax in which the value is entered for this setting is: -Replace @{ AdditionalOwners = "Owner4","Owner5"} -Replace @{ AcceptMessagesOnlyFrom = "User4","User5"} -Replace @{ AcceptMessagesOnlyFromGroups = "Group4","Group5"} -Replace @{ RejectMessagesFrom = "User4","User5"} -Replace @{ AcceptMessagesOnlyFrom = "Group4","Group5"} As the value of replacing objects, the setting accepts all the identities supported by the AdditionalOwners parameter, which is the distinguished name (DN), globally unique identifier (GUID) or samAccountName of the user, contact, or security group.
	Set-SmartGroup, **Convert-Group**, **Set-Dynasty** This setting applies to the following multi-valued parameters and lets you entirely overwrite all of

Parameter Name	Description	
	their existing values. Parameters and the syntax of their values follows:	
	Parameter	**Syntax**
	SearchContainers (StartPaths can be used as an alternative name of this parameter for this setting)	-Replace @{ SearchContainers = "Container1#1"} Or -Replace @{ StartPaths = "Container1#1"} # specifies the search scope for each container. If scope is not given, then subtree (2) is used as default.
	IncludeRecipients (Includes can be used as an alternative name of this parameter for this setting)	-Replace @{ IncludeRecipients = "Object3","Object4"} Or -Replace @{ Includes = "Object3","Object4"}
	ExcludeRecipients (Excludes can be used as an alternative name of this parameter for this setting)	-Replace @{ ExcludeRecipients = "Object5"} Or -Replace @{ Excludes = "Object5"}
	AdditionalOwners	-Replace @{ AdditionalOwners = "Owner4","Owner5"}
Only Set-Dynasty has this attribute.	**GroupBy**	Replace-Remove @{Groupby=title}
	AcceptMessagesOnlyFrom (AuthOrig can be used as an alternative name of this parameter for this setting)	-Replace @{ AcceptMessagesOnlyFrom = "User4","User5""} Or -Replace @{ AuthOrig = "User4","User5"}
	AcceptMessagesOnlyFromGroups	-Replace @{ AcceptMessagesOnlyF

Parameter Name	Description	
	(DLMemSubmitPerms can be used as an alternative name of this parameter for this setting)	romGroup = "Group4","Group5"} Or -Replace @{ DLMemSubmitPerms = "Group4","Group5"}
	RejectMessagesFrom (UnauthOrig can be used as an alternative name of this parameter for this setting)	-Replace @{ RejectMessagesFrom = "User4","User5"} Or -Replace @{ UnauthOrig = "User4","User5"}
	RejectMessagesFromGroup (DLMemRejectPerms can be used as an alternative name of this parameter for this setting)	-Replace @{ RejectMessagesFromGroup = "Group4","Group5"} Or -Replace @{ DLMemRejectPerms = "Group4","Group5"}
	The setting accepts all of the identities supported by the parameter as the value of objects for each parameter. For example, for the SearchContainer parameter, the setting can accept the DN and GUID of the domains or containers being searched for group members.	
ReportToManagerEnabled	Specify True to send non-delivery reports to the group owner or manager. The default value is False.	
ReportToOriginatorEnabled	Specify True to send non-delivery reports to the message originator. The default value is False.	
RestoreReplication	It will start the restore replication process.	
RoleCriteriaDN	Specify the criteria for a role. The criteria can be a group or a container. • Group - users that are members of the specified group will be assigned this role. • Container - users who reside in the specified container will be assigned this role.	
RoleCriteriaFilters	Specifies the filter criteria for a role. Values to this parameter are supplied as a 3-length array. • The first index contains the filter name which can be one of the 'name' or 'type'	

Parameter Name	Description
	representing 'client name' and 'client type' respectively. • The second index contains the operator which can be either 'is exactly' or 'is not'. • The third index contains the value. It can either be the client type or client name, depending on the value in the first index. Example: @('name', 'is exactly', 'automate arslanahmadvm') is a valid filter criteria. However, @('client type', 'is not', 'managementshell') is not valid because the value at first index is not correct.
RoleCriteriaOperator	Specify the operator for criteria filters of a role. The operators can be *And* or *Or*
RoleCriteriaScope	Specify the scope for a role. This parameter can be used in conjunction with *RoleCriteriaDN* to change the role criteria scope from container to group and vice-versa.
RoleDescription	Description of an identity store security role.
RoleDisabled	If a new role is created using the Set-IdentityStore commandlet, the role is created as disabled in the identity store.
RoleName	Name of an identity store security role.
RoleNameToCopy	While creating a new role, specify the name of a role you want to make a copy of. The new role is created using the settings of this role.
RoleOperation	While modifying an identity store settings using the Set-IdentityStore commandlet, specify the action to perform on an identity store security role. Possible actions are: • Add • Remove • Remove all
RolePermissions	While modifying an identity store settings using the Set-IdentityStore commandlet, specify the permission(s) that are to be granted or denied to the security role.
RolePriority	Set a role priority by specifying a value in the range of 1-99. Role priority determines which role is higher than the other, where 1 indicates the highest priority and 99 indicates the lowest priority.

Parameter Name	Description
RoleReadonly	While modifying an identity store using the Set-IdentityStore commandlet, specify that the role is created as read-only.
RoleSystemOnly	While modifying an identity store using the Set-IdentityStore commandlet, specify that the role is created as system only.
S	
SamAccountName	The logon name for the pre-Windows 2000 versions of operating systems. The value is limited to 24 characters only.
ScheduleName	Name of a schedule job to identify a schedule. The schedule job is displayed with this name against the Scheduling node on GroupID Management Console.
Script	The Smart Script for memberships update. The script should be written in Visual Basic .NET in a format recognized by Group Script Editor. Imanami recommends writing the script in a separate file, instead of writing the complete script on the shell, and give the path of the script file using the ScriptFilePath setting. **Note:** If while writing script using this setting, you must use a parameter's value that is enclosed in double-quotes (""), insert an apostrophe (') before every quotation mark. For example, #Region ' "Imanami Generated Code' ".
ScriptFilePath	The path to the script file containing Smart Script for memberships update. The script should be written in Visual Basic .NET (having .vb extension) in a format recognized by Group Script Editor.
SearchContainer	The distinguished name (DN) or globally unique identifier (GUID) of the domain or one or more containers in which to search for users, contact, group or dynasty members.
SearchContainersScopeList	This setting works in conjunction with the SearchContainer setting and sets the scope for the object search. Following are the possible values for this parameter: • 1 (Limits search to the container specified in the SearchContainer parameter and ignores the sub-containers.) • 2 (Searches the whole sub-tree, including the base container specified in the

Parameter Name	Description
	SearchContainer parameter and all its sub-containers. This is also the default setting for this parameter; therefore, if the search scope is not explicitly specified, this value is used.) Although the values are numerical, you must enclose them in double-quotes. For example: "1", "2".
SecurityQuestions	Adds a security question in an identity store.
SecurityToken	When you the Get-Token command, you get a value against Claims. Provide that value to this parameter.
SecurityType	The access level of the group: Private, Semi_Private and Public. If this parameter is not given, the group is created as Private.
SendEmail	Specify this parameter to send the password expiry e-mail notifications. The group must have an e-mail address and notifications must be configured. (Applies to Password Expiry group)
SendOofMessageToOriginatorEnabled	Specify True to enable the group to send Out-of-Office messages to e-mail senders. The default value is False.
SendToOwners	Sends job completion notifications to group owners and additional owners as well as to the other specified recipients.
Separator	Specifies a character to use in the display name and the alias to separate group-by values from the each other.
Server	The server name for the following data sources, if specified: • Microsoft SQL Driver (Name of the Microsoft SQL Server that contains the database you want to connect to) • Oracle (Name of the Oracle server that contains the database you want to connect to) • Lotus Notes (Name of the Lotus Notes server that contains the database you want to connect to) • Sun ONE iPlanet Driver (DNS name or IP address of SunONE server)
SetNotifications	Enables or disables notifications for a scheduled job.

Parameter Name	Description
ShouldReturnCollection	Specifying this parameter returns a single collection of objects containing all groups.
SimpleDisplayName	The printable display name for an object. The printable display name is usually the combination of the user's first name, middle initial, and last name.
SmartDLNotes	The notes entered here are copied to all smart groups created using Automate.
SmartFilter	Adds a smart search filter that applies only on SmartGroups. The smart search filters are: • IsExpired • GroupExpiringIn • SecurityType • ExpirationPolicy. (Functional in **Get-Group** and **Get-Smartgroup** commandlets only)
SmartGroupType	The type of Smart Group that you want the commandlet to retrieve. Values of this parameter are: • SmartGroup • SmartDynasty Omitting this parameter retrieves both SmartGroups and Dynasties.
SmsGatewayName	The name of an SMS gateway.
SmtpPassword	This setting works in conjunction with the UseSmtpUserAuthentication, SmtpServer, SmtpUserName, SmtpPort and SmtpSSLEnabled settings and sets the password of the user account to be used for communicating with an external SMTP server.
SmtpPort	This setting works in conjunction with the SmtpServer setting and sets the port number to be used for communicating with an SMTP server.
SmtpServer	The fully qualified name or IP address of an SMTP server. GroupID will route messages through this server.
SmtpSSLEnabled	This setting works in conjunction with the SmtpServer, SmtpUserName, SmtpPassword, and SmtpPort settings. Enter True if the external SMTP server is SSL-enabled.
SmtpUserName	This setting works in conjunction with the UseSmtpUserAuthentication, SmtpServer, SmtpPassword and SmtpPort settings and sets the

Parameter Name	Description
	e-mail address of the user account to be used for communicating with an external SMTP server.
SQLServer	SQL server name on which database of previous GroupID version is hosted.
StartDate	Date from which membership will be started or revoked or removed.
SslEnabled	Specify that the SMTP server is SSL enabled.
StartTime	Time of the day at which the schedule is triggered.
State	The state for a user, contact or mailbox.
Storage	Filters the mailboxes to be returned. If specified, only mailboxes on the specified server or mailbox store (Exchange 2007-SP3 and later/2010/2013/2016) will be returned. Custom recipients, public folders and distribution lists are not affected by this filter. Typing an asterisk (*) as a value of this parameter searches all mailboxes on any server.
StoreDescription	Description of an identity store.
StoreEnabled	Enables or disables an identity store.
Subject	The subject of the e-mail notification.
Sun_Container	Specify distinguished name (DN) of a container in an external datasource (specifically Sun ONE iPlanet datasource) in Query Designer of a smart group.
SupportEmail	The e-mail address of the group or contact providing support to users of Password Center and Self-Service portals. This support e-mail address is set by default for all new portals created using Password Center and Self-Service.
SupportURL	The default URL of the online help for Self-Service portals. This URL is set by default for all new portals created using Self-Service.
SWAMobileAttribute	The name of the attribute used by Second Way Authentication via mobile.
SWAQuestions	The question for security questions based Second Way Authentication. The value to this parameter is supplied as 2-length arrays. The first index contains the question text and the second index contains the name of the attribute for that question.

Parameter Name	Description
SWAQuestionsOperation	The action to perform on the SWAQuestions parameter.
SWAuthenticationFactor	Number of authentication types enforced for the security role.
SWEmailAttribute	The name of the attribute used by Second Way Authentication via email.
SystemDSN	The System Data Source Name (DSN) to use as the data source, if the value of the DataSourceType parameter is ODBC Data Source.
T	
TableorView	The table or view name if the value of the DataSourceType parameter is ODBC Data Source, Microsoft SQL Driver or Oracle.
TargetOperation	The actions to perform on targets. Possible values are: • Add • Remove
Targets	Provide the names of groups and containers the job will process as per the action provided in the *TargetOperation* parameter.
TemplateFile	The path of the template file that the commandlet should use for generating the e-mail contents.
Title	Title of a user, contact or mailbox.
ToEmail	Recipient of the email notification.
TopLevelOnly	Sets whether the search should return matches only from top-level dynasties or includes sub-level dynasties in the search as well. The default value 0 (zero) returns results from the complete hierarchy of dynasties. Specify the value 1 to return matches from only top-level dynasties.
TopManager	The distinguished name (DN), globally unique identifier (GUID) or samAccountName of the top-level manager. The commandlet constructs a Managerial Dynasty structure by creating a Smart Group for all direct reports to the selected top-level manager and continues down the Dynasty structure by creating SmartGroups for all direct reports to sublevel managers. (Applies to Managerial Dynasty)

Parameter Name	Description
TriggerId	Unique identity of a trigger. The ID can be retrieved from the Triggers property of *Get-Schedule* commandlet.
TriggerOperation	The actions to perform on the provided triggers . Possible actions are: • Add • Remove single by id • Remove by type • remove all
TriggerType	The trigger type while adding or removing triggers to/from a schedule. This parameter is also used to select a schedule with a particular trigger type. Possible trigger types are: • RunOnce • RunDaily • RunWeekly • RunMonthly • RunMonthlyDOW • OnIdle • OnSystemStart • OnLogon
Type	**New-Group, New-Dynasty, New-SmartGroup, New-Dynasty, Convert-Group** Specifies that the new group or dynasty will be used either for mail distribution (a Distribution group or dynasty) or for securing public folders or other resources (a Security group or Dynasty). **Set-Group, Set-SmartGroup, Set-Dynasty** The type of the group to be modified. The available types are: Distribution and Security. **Add-GroupMember** Perpetual, Temporary Member or Addition Pending **Remove-GroupMember** Removal Pending, Temporary Removed. If no type is given then it will be considered Perpetual remove.
U	
UpdateChildren	The default value True forces Automate to update the children of a Dynasty when it updates the Dynasty itself. Set its value to False to disable this feature.

Parameter Name	Description
UpdateMembershipByManagerEnabled	A True value enables the group manager to update the group membership list. The default value is False.
Username	The name of the user that will be used for the execution of the commandlet in which it is mentioned. This parameter and the Credentials parameter cannot be used simultaneously in a commandlet.
UseSmtpUserAuthentication	Set its value to True to use SMTP authentication for communicating with the SMTP server. The default value is False. The authentication details are provided by the SmtpUserName, SmtpPassword, SmtpPort and SmtpSSLEnabled settings.
UseSmtpUserAuthentication	Specify if user authentication of SMTP server is to be used.
V	
ValidationDateRemovalInterval	Specify the number of days since the last profile validation date. GroupID clears the validation date and the policies for new users are applied to this user.
Verb	Shows information about commandlets or command elements having the specified verb in their name. Wildcard search is also supported.
W	
Weekdays	Specify the weekdays for the weekly triggers. Possible values are: • Sunday • Monday • Tuesday • Wednesday • Thursday • Friday • Saturday • AllDays
WeeksInterval	Specify weekly interval in weekly triggers i.e. number of weeks after which a scheduled job is repeated.
WhenGroupMembershipThresholdReach	Policy to apply when membership change threshold, specified in out-of-bounds configurations, is reached. Possible values are: • PreventUpdation

Parameter Name	Description
	• NestIntoChildGroups
WindowsAuthentication	Enables Windows Authentication mode for SQL Server. In Windows Authentication mode, administrators can enable users to log on to the SQL Server using their Windows credentials.
WindowsLoggingEvent	Set events for logging from all GroupID modules in a centralized event log named Imanami GroupID that can be viewed from the Windows Event Viewer. Possible events are: • FailureAudit • SuccessAudit • Info • Warn • Error
X	
XDaysBeforeLeaveNotificationMB	Specify the number of days. The temporary additional owner / manager of a group receives a notification before the specified number of days he or she is removed as additional owner / manager.
XDaysBeforeLeaveNotificationML	Specify the number of days. The user receives a notification before the specified number of days he or she is removed from a group membership.
Y	
YearMonths	Specify the months of years for monthly triggers. Possible values are: • January • February • March • April • May • June • July • August • September • October • November • December • AllMonths
Z	
Zip	The zip code for a user, contact or mailbox.

Common Parameters

GroupID Management Shell does not support Windows Powershell parameter **–CommonParameter** in its commandlets. It refers to the following common patameters:

- Confirm
- Debug
- ErrorAction
- ErrorVariable
- InformationAction
- InformationVariable
- OutBuffer
- OutVariable
- PipelineVariable
- Verbose
- WarningAction
- WarningVariable
- WhatIf
- Write-Information

See About Common Parameters for further details.

Appendix A

Setting the $Credentials environment variable

By default, the GroupID Management Shell uses the credentials of current user logged-on to the identity store for executing commandlets. If you need to use a different user account for some commandlets, you must set the **$Credentials** environment variable to the credentials of that user. This user account must also be part of the same forest. Once set, the variable can be used as a value for the **Credential** parameter with those commandlets that you want to execute using this account. The rest of the commandlets are executed under the credentials of the local user.

Syntax

```
$Credentials = new-object
System.Management.Automation.PsCredential "DomainName\User
Name",$(convertto-securestring "Password" -asplaintext -
force)
```

Example 1

The following command sets the **$Credentials** environment variable to the credentials of the user, **John Smith**, which exists on the same domain you are logged-on to.

```
$Credentials = new-object
System.Management.Automation.PsCredential
"JohnSmith",$(convertto-securestring "MyP@ssw0rd" -
asplaintext -force)
```

Example 2

The command below sets the credentials of the user, **Brian Regan**, which exists on a different domain on the same forest.

```
$Credentials = new-object
System.Management.Automation.PsCredential
"Sales.Imanami.US\BrianRegan",$(convertto-securestring
"MyP@ssw0rd" -asplaintext -force)
```

Example 3

The following command shows how to use the **$Credentials** environment variable with commandlets.

```
New-Container -ParentContainer "DC=HR,DC=Imanami,DC=US" -
Name "Recruiting" -Credential $Cred
```


Imanami Corporation

2301 Armstrong Street
Livermore, CA 94551
United States
https://www.imanami.com/

Support: (925) 371-3000, Opt. 3
support@imanami.com

Sales: (925) 371-3000, Opt. 1
sales@imanami.com

Toll-Free: (800) 684-8515
Phone: (925) 371-3000
Fax: (925) 371-3001