5

Distribution Groups and Address Lists

In this chapter, we will cover the following:

* Reporting on distribution group membership
* Adding members to a distribution group from an external file
* Previewing dynamic distribution group membership
* Excluding hidden recipients from a dynamic distribution group
* Converting and upgrading distribution groups
* Allowing managers to modify group permissions
* Removing disabled users from distribution groups
* Working with distribution group naming policies
* Working with distribution group membership approval
* Creating address lists
* Exporting address list membership to a CSV file
* Configuring hierarchical address books

# Introduction

In Chapter 3, we looked at managing recipients, which covered the process of creating and modifying the membership of both regular and dynamic distribution groups. In this chapter, we are going to dive deeper into distribution group management within the Exchange Management Shell. The recipes in this chapter provide solutions to some of the most common distribution group management tasks that can, and sometimes must, be handled from the command line. Some of the topics we'll cover include the implementation of group naming policies, allowing group managers to modify the memberships of distribution groups, and more. We'll also go over the process of some basic address list management that can be automated through the shell.

## Performing some basic steps

To work with the code samples in this chapter, follow these steps to launch the Exchange Management Shell:

1. Log onto a workstation or server with the Exchange Management Tools installed.
2. You can connect using remote PowerShell if you for some reason don't have Exchange Management Tools installed. Use the following command:

$Session = New-PSSession -ConfigurationName Microsoft.Exchange `

-ConnectionUri http://tlex01/PowerShell/ `

-Authentication Kerberos

Import-PSSession $Session

1. Open the Exchange Management Shell by clicking the windows button and go to Microsoft Exchange Server 2016 | Exchange Management Shell.

If any additional steps are required they will be listed at the beginning of the recipe in the Getting ready section.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Remember to start the Exchange Management Shell using **Run as Admin** to avoid permission problems.  In the chapter, notice that in the examples of cmdlets, I have used the back tick (`) character for breaking up long commands into multiple lines.  The purpose with this is to make it easier to read. The back ticks are not required and should only be used if needed.  Notice that the Exchange variables, like $exscripts, are not available when using the method above. |  |

# Reporting on distribution group membership

One of the common requests you are likely to receive as an Exchange administrator is to generate a report detailing which recipients are members of one or more distribution groups. In this recipe, we'll take a look at how to retrieve this information from the Exchange Management Shell.

## How to do it...

To view a list of each distribution group member interactively, use the following code:

foreach($i in Get-DistributionGroup -ResultSize Unlimited) {

Get-DistributionGroupMember $i -ResultSize Unlimited |

Select-Object @{n="Member";e={$\_.Name}},

RecipientType,

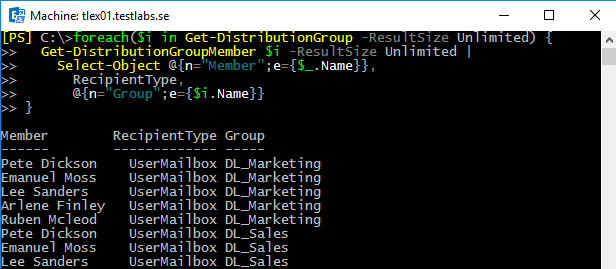
@{n="Group";e={$i.Name}}

}

This will generate a list of Exchange recipients and their associated distribution group membership.

## How it works...

This code loops through each item returned from the Get-DistributionGroup cmdlet.   
As we process each group, we run the Get-DistributionGroupMember cmdlet to determine the member list for each group and then use Select-Object to construct a custom object that provides Member, RecipientType, and Group properties. Notice that, when running both Exchange cmdlets, we're setting the -ResultSize parameter to Unlimited to ensure that the details will be retrieved in the event that there are more than 1,000 groups or group members. The result of the cmdlets above will look similar like the screenshot below:



7081EN\_05\_01

## There's more...

The previous code sample will allow you to view the output in the shell. If you want to export this information to a CSV file, use the following code:

$report=foreach($i in Get-DistributionGroup -ResultSize Unlimited) {

Get-DistributionGroupMember $i -ResultSize Unlimited |

Select-Object @{n="Member";e={$\_.Name}},

RecipientType,

@{n="Group";e={$i.Name}}

}

$report | Export-CSV c:\GroupMembers.csv –NoType

The difference this time is that the output from our code is being saved in the $report variable. Once the report has been generated, the $report object is then exported to   
a CSV file that can be opened in Excel.

## See also

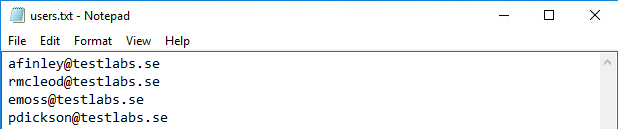
* Previewing dynamic distribution group membership
* Adding members to a distribution group from an external file

# Adding members to a distribution group from an external file

When working in large or complex environments, performing bulk operations is the key to efficiency. Using PowerShell core cmdlets such as Get-Content and Import-CSV, we can easily import external data into the shell and use this information to perform bulk operations on hundreds or thousands of objects in a matter of seconds. Obviously, this can vastly speed up the time we spend on routine tasks and greatly increase our efficiency. In this recipe, we'll use these concepts to add members to distribution groups in bulk from a text or CSV file using the Exchange Management Shell.

## How to do it...

1. Create a text file called c:\Scripts\users.txt that lists the recipients in your organization that you want to add to a group. Make sure you enter them one line at a time, as shown in the following screen shot:



7081EN\_05\_02

1. Next, execute the following code to add the list of recipients to a distribution group:

Get-Content c:\Scripts\users.txt | ForEach-Object {

Add-DistributionGroupMember –Identity Sales -Member $\_

}

When the code runs, each user listed in the c:\Scripts\users.txt file will be added to the Sales distribution group.

## How it works...

When importing data from a plain text file, we use the Get-Content cmdlet, which will   
read the content of the file into the shell one line at a time. In this example, we pipe the content of the file to the ForEach-Object cmdlet, and as each line is processed we   
execute the Add-DistributionGroupMember cmdlet.

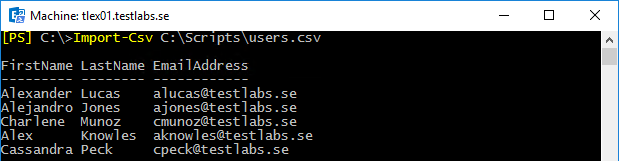
Inside the ForEach-Object script block we use the Add-DistributionGroupMember cmdlet and assign the $\_ object, which is the current recipient item in the pipeline, to   
the -Member parameter.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | To remove recipients from a distribution group, you can use the  Remove-DistributionGroupMember cmdlet. |  |

Keep in mind that this text file does not have to contain the SMTP address of the recipient. You can also use the Active Directory account name, User Principal Name, Distinguished Name, GUID, or LegacyExchangeDN values. The important thing is that the file contains a valid and unique value for each recipient. If the identity of the recipient cannot be determined, the Add-DistributionGroupMember command will fail.

## There's more...

In addition to using plain text files, we can also import recipients from a CSV file and add them to a distribution group. Let's say that you have a CSV file setup with multiple columns, such as FirstName, LastName, and EmailAddress. When you import the CSV file, the data can be accessed using the column headers as the property names for each object. Take a look at the following screenshot:



7081EN\_05\_03

Here you can see that each item in this collection has an EmailAddress property. As long as this information corresponds to the recipient data in the Exchange organization, we can simply loop through each record in the CSV file and add these recipients to a distribution group:

Import-Csv C:\Scripts\users.csv | ForEach-Object {

Add-DistributionGroupMember Sales -Member $\_.EmailAddress

}

The given code uses the Import-Csv cmdlet to loop through each item in the collection. As we process each record, we add the recipient to the Sales distribution group using the $\_.EmailAddress object.

## See also

Managing distribution groups in Chapter 3, Managing Recipients

# Previewing dynamic distribution group membership

The concept of the dynamic distribution group was introduced with the initial release of Exchange 2007 and included a new way to create and manage distribution groups. Unlike regular distribution groups whose members are statically defined, a dynamic distribution group determines its members based on a recipient filter. These recipient filters can be very complex, or they can be based on simple conditions, such as including all the users with a common value set for their Company or Department attributes in Active Directory. Since these dynamic groups are based on a query, they do not actually contain group members and if you want to preview the results of the groups query in the shell you need to use a series of commands. In this recipe, we'll take a look at how to view the membership of dynamic distribution groups in the Exchange Management Shell.

## How to do it...

Imagine that we have a dynamic distribution group named Legal that includes all of the users in Active Directory with a Department attribute set to the word Legal. We can use   
the following commands to retrieve the current list of recipients for this group:

$legal= Get-DynamicDistributionGroup -Identity legal

Get-Recipient -RecipientPreviewFilter $legal.RecipientFilter

## How it works...

Recipient filters for dynamic distribution groups use OPATH filters that are accessible through the RecipientFilter property of a dynamic distribution group object. As you can see here, we have specified the Legal groups OPATH filter when running the Get-Recipient cmdlet with the –RecipientPreviewFilter parameter. Conceptually, this would be similar to running the following command:

Get-Recipient -RecipientPreviewFilter "Department -eq 'Legal'"

Technically, there is a little bit more to it than that. If we were to actually look at the value for the RecipientFilter property of this dynamic distribution group, we would see much more information in addition to the filter defined for the Legal department. This is because Exchange automatically adds several additional filters when it creates a dynamic distribution group that excludes system mailboxes, discovery mailboxes, arbitration mailboxes, and more. This ends up being quite a bit of information, and creating an object instance of the dynamic distribution group gives us easy access to the existing OPATH filter that can be previewed with the Get-Recipient cmdlet.

## There's more...

When working with regular distribution groups, you may notice that there is a cmdlet called Get-DistributionGroupMember. This allows you to retrieve a list of every user that is a member of a distribution group. Unfortunately, there is no equivalent cmdlet for dynamic distribution groups, and we need to use the method outlined previously that uses the Get-Recipient cmdlet to determine the list of recipients in a dynamic distribution group.

If you find yourself doing this frequently, it probably makes sense to wrap these commands up into a function that can be added to your PowerShell profile. This will allow you to determine the members of dynamic distribution group using a single command that will be available to you every time you start the shell. Here is the code for a function called Get-DynamicDistributionGroupMember, which can be used to determine the list of recipients included in a dynamic distribution group:

function Get-DynamicDistributionGroupMember {

param(

[Parameter(Mandatory=$true)]

$Identity

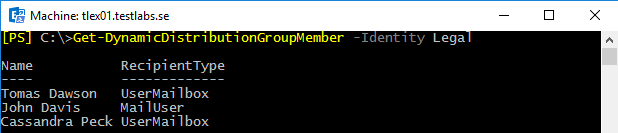
)

$group = Get-DynamicDistributionGroup -Identity $Identity

Get-Recipient -RecipientPreviewFilter $group.RecipientFilter

}

Once this function is loaded into your shell session, you can run it just like a cmdlet:



7081EN\_05\_04

You can see that the command returns the recipients that match the OPATH filter for the Legal distribution group and is much easier to type than the original example.

## See also

* Reporting on distribution group membership
* Working with recipient filters in Chapter 3, Managing Recipients

# Excluding hidden recipients from a dynamic distribution group

When creating dynamic distribution groups through the Exchange Management Console, you can specify which recipients should be included in the group using a basic set of conditions. If you want to do more advanced filtering, such as excluding hidden recipients, you will need to configure OPATH filters for your dynamic distribution groups through the Exchange Management Shell. In this recipe, you'll learn how to use the shell to create a recipient filter that excludes hidden recipients from dynamic distribution groups.

## How to do it...

Let's say that we need to set up a distribution group for our TechSupport department. The following commands can be used to create a dynamic distribution group that includes all the mailboxes for the users in the TechSupport OU that are not hidden from address lists:

New-DynamicDistributionGroup -Name TechSupport `

-RecipientContainer contoso.com/TechSupport `

-RecipientFilter {

HiddenFromAddressListsEnabled -ne $true

}

## How it works...

When you want to exclude a mailbox, contact, or distribution group from an address list, you set the HiddenFromAddressListsEnabled property of the recipient to $true. This is often done for special purpose recipients that are used for applications or services that should not be visible by the general end-user population. While this takes care of address lists, it does not affect your dynamic distribution groups, and if you want to exclude these recipients you'll need to use a similar filter to the one shown in the previous example. When we created the TechSupport dynamic distribution group, we used a very basic configuration that included all the recipients that exist within the TechSupport OU in Active Directory. Our custom recipient filter specifies that the HiddenFromAddressListEnabled property of each recipient can’t be equal to $true. With this filter in place, only recipients that are not hidden from Exchange address lists are included as dynamic distribution group members.

Keep in mind that, when you create a dynamic group using the –RecipientFilter parameter, any future changes will have to be made through the Exchange Management   
Shell. If you need to change the recipient filter at any time, you cannot use Exchange Admin Center and will need to use the Set-DynamicDistributionGroup   
cmdlet to make the change.

## There's more...

Updating a recipient filter for an existing dynamic distribution group can be a bit tricky.   
This is because the recipient filters are automatically updated by Exchange to exclude   
certain types of resource and system mailboxes. Let's go through the process of creating   
a new dynamic distribution group, and then we'll modify the recipient filter after the fact so that you can understand how this process works.

First, we'll create a new dynamic distribution group for the Marketing department using a basic filter. Only users with e-mail addresses that contain the word Marketing will be members of this group:

New-DynamicDistributionGroup -Name Marketing `

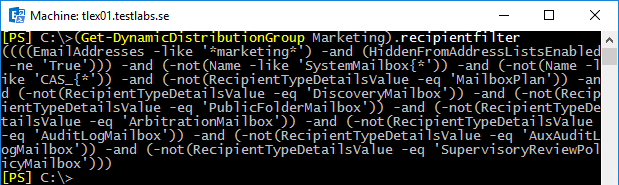
-RecipientContainer contoso.com/Marketing `

-RecipientFilter {

EmailAddresses -like '\*marketing\*'

}

Now that the group has been created, let's verify the recipient filter by accessing the RecipientFilter property of that object:



7081EN\_05\_05

As you can see from the output, we get a lot more back than we originally put in. This is how Exchange prevents the dynamic distribution groups from displaying recipients such as system and discovery mailboxes in your dynamic distribution lists. You do not need to worry about this extraneous code when you update your filters, as it will automatically be added back in for you when you change the recipient filter.

Now that we understand what's going on here, let's update this group so that we can also exclude hidden recipients. To do this, we need to construct a new filter and use the   
Set-DynamicDistributionGroup cmdlet as shown next:

Set-DynamicDistributionGroup -Identity Marketing `

-RecipientFilter {

(EmailAddresses -like '\*marketing\*') -and

(HiddenFromAddressListsEnabled -ne $true)

}

Using this command, we've specified the previously-configured filter in addition to the new one that excludes hidden recipients. In order for recipients to show up in this dynamic distribution group, they must have the word Marketing somewhere in their e-mail address and their account must not be hidden from address lists.

## See also

* Hiding recipients from address lists in Chapter 3, Managing Recipients
* Working with recipient filters in Chapter 3, Managing Recipients

# Converting and upgrading distribution groups

If you for some reason haven’t upgraded your legacy distribution groups, this recipe is for you because before when migrating to Exchange 2010 from Exchange 2003, you may be carrying over several mail-enabled non-universal groups. These groups will still function, but the administration of these objects within the Exchange tools will be limited. In addition, several distribution group features provided by Exchange 2010, 2013 or 2016 will not be enabled for a group until it has been upgraded. If you somehow haven’t upgraded your groups before, it’s about time now. That’s why the recipe is included in this book. This recipe covers the process of converting and upgrading these groups within the Exchange Management Shell.

## How to do it...

1. To convert all of your non-universal distribution groups to universal, use the following one-liner:

Get-DistributionGroup -ResultSize Unlimited `

-RecipientTypeDetails MailNonUniversalGroup |

Set-Group -Universal

1. Once all of your distribution groups have been converted to universal, you can upgrade them using the following command:

Get-DistributionGroup –ResultSize Unlimited |

Set-DistributionGroup –ForceUpgrade

## How it works...

The first command will retrieve all the non-universal mail-enabled distribution groups in your organization and pipe the results to the Set-Group cmdlet which will convert them using the -Universal switch parameter. It may not be a big deal to modify a handful of groups, but if you have hundreds of mail-enabled non-universal groups the command in the previous example can save you a lot of time.

If you have a large number of groups to convert, you may find that some of them are members of another global group and cannot be converted. Keep in mind that a universal group cannot be a member of a global group. If you run into errors because of this, you can convert these groups individually using the Set-Group cmdlet. Then you can run the command in the previous example again to convert any remaining groups in bulk.

Even after converting non-universal groups to universal, you'll notice that, when viewing the properties of a distribution group created by Exchange 2003, you cannot manage things such as message moderation and membership approval. In order to fully manage   
these groups, you need to upgrade them using the -ForceUpgrade parameter with the   
Set-DistributionGroup cmdlet. Keep in mind that after the upgrade these objects can no longer be managed using anything other than the Exchange 2010, 2013 or 2016 management tools.

## There's more...

The Exchange Management tools, both the graphical console and the shell, can only be   
used to create distribution groups using universal group scope. Additionally, you can only   
mail-enable existing groups with universal group scope. If you've recently introduced   
Exchange into your environment, you can convert existing non-universal, non-mail   
enabled groups in bulk using a one-liner:

Get-Group –ResultSize Unlimited `

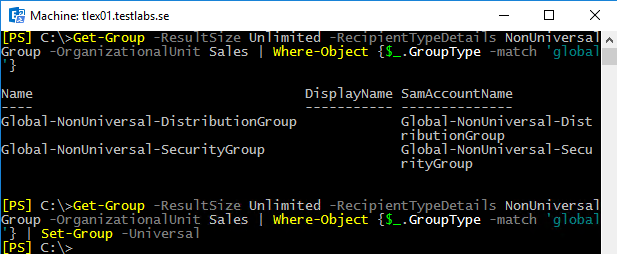
-RecipientTypeDetails NonUniversalGroup `

–OrganizationalUnit Sales |

Where-Object {$\_.GroupType -match 'global'} |

Set-Group -Universal

As you can see in this example, we are retrieving all non-mail enabled, non-universal global groups from the Sales OU and converting them to universal in a single command. See the screenshot below for the outcome of the cmdlets above:



7081EN\_05\_06

You can change the OU or use additional conditions in your filter based on your needs. Once the group is converted it can be mail-enabled using the Enable-Distribution group cmdlet and it will show up in the list of available groups when creating new distribution groups.

## See also

Managing distribution groups

Allowing managers to modify group membership

# Allowing managers to modify group membership

Many organizations like to give specific users rights to manage the membership of designated distribution groups. This has been a common practice for years in previous versions of Exchange. While users have typically modified the memberships of the groups they have rights to from within Outlook, they now have the added capability to manage these groups from the web-based Exchange Control Panel (ECP). Exchange 2010 introduced a new security model that changed the way you can delegated these rights. In this recipe, we'll take a look at what you need to do in Exchange 2016 to allow managers to modify the memberships of distribution groups.

## How to do it...

1. The first thing you need to do is assign the built-in MyDistributionGroups role to the Default Role Assignment Policy:

New-ManagementRoleAssignment -Role MyDistributionGroups `

-Policy "Default Role Assignment Policy"

1. Next, set the ManagedBy property of the distribution group that needs to be modified:

Set-DistributionGroup Sales -ManagedBy bobsmith

After running the given command, Bob Smith has the ability to modify the membership of the Sales distribution group through ECP, Outlook, or the Exchange Management Shell.

## How it works...

In order to allow managers to modify the membership of a group, we need to do some initial configuration through the Exchange 2016 security model called Role Based Access Control (RBAC). The MyDistributionGroups role is an RBAC management role that allows end-users to view, remove, and add members to distribution groups where they have been added to the ManagedBy property.

By default, the MyDistribitionGroups management role is not assigned to anyone. In the first step, we added this role to the Default Role Assignment Policy that is assigned to all users by default.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | In addition to using the shell, you can assign the MyDistributionGroups management role to the Default Role Assignment Policy using ECP. |  |

In the next step, we assigned a user to the ManagedBy property of the Sales distribution group. The ManagedBy attribute is a multi-valued property that will accept multiple users if you need to allow several people to manage a distribution group.

The reason that the MyDistributionGroups role is not enabled by default is because, in addition to allowing users to modify the groups that they own, it also allows them to create new distribution groups from within the ECP. While some organizations may like this feature, others may not be able to allow this since the provisioning of groups may need to be tightly controlled. Make sure you keep this in mind before implementing this solution.

## There's more...

If you need to prevent users from creating their own distribution groups, then you do not want to assign the MyDistributionGroups role. Instead, you'll need to create a custom RBAC role. This can only be accomplished using the Exchange Management Shell.

To implement a custom RBAC role that will only allow users to modify distribution groups that they own, we need to perform a few steps. The first thing we need to do is create a child role based on the existing MyDistributionGroups management role, as shown next:

New-ManagementRole -Name MyDGCustom -Parent MyDistributionGroups

After running this command, we should now have a new role called MyDGCustom that contains all of the cmdlets that will allow the user to add and remove distribution groups. Using the following commands, we'll remove those cmdlets from the role:

Remove-ManagementRoleEntry MyDGCustom\New-DistributionGroup

Remove-ManagementRoleEntry MyDGCustom\Remove-DistributionGroup

This modifies the role so that only the cmdlets that can get, add, or remove distribution group members are available to the users.

Finally, we can assign the custom role to the Default Role Assignment Policy, which, out of the box, is already applied to every mailbox in the organization:

New-ManagementRoleAssignment -Role MyDGCustom `

-Policy "Default Role Assignment Policy"

Now that this custom RBAC role has been implemented, we can simply add users to the ManagedBy property of any distribution group and they will be able to add members to and remove members from that group. However, they will be unable to delete the group, or create a new distribution group, which accomplishes the goal.

## See also

* Working with Role Based Access Control (RBAC) in Chapter 10, Exchange Security
* Troubleshooting Role Based Access Control in Chapter 10, Exchange Security

# Removing disabled user accounts from distribution groups

A standard practice amongst most organizations when users leave or have been let go is to disable their associated Active Directory user account. This allows an administrator to easily re-enable the account in the event that the user comes back to work, or if someone else needs access to the account. Obviously, this has become a common practice because the process of restoring a deleted Active Directory user account is a much more complex alternative. Additionally, if these user accounts are left mailbox-enabled, you can end up with distribution groups that contain multiple disabled user accounts. This recipe will show you how to remove these disabled accounts using the Exchange Management Shell.

## How to do it...

To remove disabled Active Directory user accounts from all distribution groups in the organization, use the following code:

$groups = Get-DistributionGroup -ResultSize Unlimited

foreach($group in $groups){

Get-DistributionGroupMember $group |

?{$\_.RecipientTypeDetails -like '\*User\*'} |

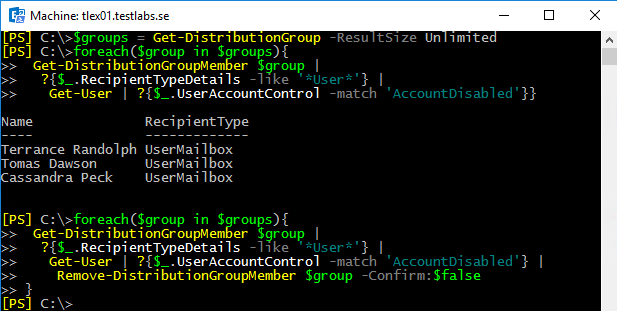
Get-User | ?{$\_.UserAccountControl -match 'AccountDisabled'} |

Remove-DistributionGroupMember $group -Confirm:$false

}

## How it works...

This code uses a foreach loop to iterate through each distribution group in the organization. As each group is processed, we retrieve only the members whose recipient type contains the word User. We're also filtering out all non-user mailboxes as these are tied to disabled Active Directory accounts. These filters will ensure that we only pipe objects with Active Directory user accounts down to the Get-User cmdlet, which will determine whether or not the account is disabled by checking the UserAccountControl property of each object. If the account is disabled, it will be removed from the group. See the screenshot below as an example:



7081EN\_05\_07

## There's more…

Instead of performing the remove operation, we can use a slightly modified version of the previous code to simply generate a report based on disabled Active Directory accounts that are members of a specific distribution group. Use the following code to generate this report:

$groups = Get-DistributionGroup -ResultSize Unlimited

$report = foreach($group in $groups){

Get-DistributionGroupMember $group |

?{$\_.RecipientTypeDetails -like '\*User\*'} |

Get-User | ?{$\_.UserAccountControl -match 'AccountDisabled'} |

Select-Object ` Name,RecipientTypeDetails,@{n='Group';e={$group}}

}

$report | Export-CSV c:\disabled\_group\_members.csv -NoType

After running this code, a report will be generated using the specified file name that will list the disabled account name, Exchange recipient type, and associated distribution group for which it is a member.

## See also

* Managing distribution groups in Chapter 3, Managing Recipients

# Working with distribution group naming policies

Using group naming policies, you can require that the distribution group names in your organization follow a specific naming standard. For instance, you can specify that all distribution group names are prefixed with a certain word and you can block certain words from being used within group names. In this recipe, you'll learn how to work with group naming policies from within the Exchange Management Shell.

## How to do it...

To enable a group naming policy for your organization, use the   
Set-OrganizationConfig cmdlet, as shown next:

Set-OrganizationConfig -DistributionGroupNamingPolicy `

"DL\_<GroupName>"

## How it works...

Since Exchange 2010 gives your users the ability to create and manage their own distribution groups, you may want to implement a naming policy that matches your organization's naming standards. In addition, you can implement naming policies so that your administrators are required to follow a specific naming convention when creating groups.

Your distribution group naming policy can be made up of text you specify, or it can use specific attributes that map to the user who creates the distribution group. In the previous example, we specified that all distribution groups should be prefixed with DL\_ followed by the group name. The <GroupName> attribute indicates that the group name provided by the user should be used. So, if someone were to create a group named "Help Desk", Exchange would automatically configure the name of the group as DL\_Help Desk.

The following attributes can be used within your group naming policies:

* Company
* CountryCode
* CountryorRegion
* CustomAttribute1 - 15
* Department
* Office
* StateOrProvince

Title

Let's take a look at another example to see how we could implement some of these attributes within a group naming policy. Using the following command, we'll update the group naming policy to include both the Department and the State of the user creating the group:

Set-OrganizationConfig -DistributionGroupNamingPolicy `

"<Department>\_<GroupName>\_<StateOrProvince>"

Now let's say that we have an administrator named Dave who works in the IT department   
in the Arizona office. Based on this information, we know that his Department attribute   
will be set to "IT" and his State attribute will be set to "AZ". When Dave uses the   
New-DistributionGroup cmdlet to create a group for the maintenance department, specifying "Maintenance" for the –Name parameter value, Exchange will automatically apply the group naming policy, and the distribution group name will be IT\_Maintenance\_AZ.

In addition, we can exclude a list of names that can be used when creating distribution groups. This is also specified by running the Set-OrganizationConfig cmdlet. For example, to block a list of words we can use the following syntax:

Set-OrganizationConfig `

-DistributionGroupNameBlockedWordsList badword1,badword2

If a user tries to create a group using one of the blocked names, they'll receive an error that says the group name contains a word which isn't allowed in group names in your organization. Please rename your group.

There is even one more parameter to be aware about, which could be very useful.

It’s used for providing a default organization unit where the distribution groups are placed by default. See the command example:

Set-OrganizationConfig `

-DistributionGroupDefaultOU “contoso.com/Test”

This setting can be overridden by using the parameter -OrganizationalUnit when creating the distribution groups.

## There's more...

When a group naming policy is applied in your organization, it is possible to override it from within the Exchange Management Shell. Both the New-DistributionGroup and the Set-DistributionGroup cmdlets provides an -IgnoreNamingPolicy switch parameter that can be used when you are creating or modifying a group. To create a distribution group that will bypass the group naming policy, use the following syntax:

New-DistributionGroup -Name Accounting -IgnoreNamingPolicy

The graphical management tools (ECP) can be used to create distribution groups, but if a naming policy is applied to your organization and you need to override it, you must use the shell as shown previously.

You can force administrators to use group naming policies, even if they have access to   
the Exchange Management Shell. If you plan on doing this, you need to assign them to   
the New-DistributionGroup and Set-DistributionGroup cmdlets using   
a custom Role Based Access Control (RBAC) role that does not allow them to use   
the -IgnoreNamingPolicy switch parameter.

## See also

* Managing distribution groups

# Working with distribution group membership approval

You can allow end-users to request distribution group membership through the Exchange Control Panel (ECP). Additionally, you can configure your distribution groups so that users can join a group automatically without having to be approved by a group owner. We'll take a look at how to configure these options in this recipe.

## How to do it...

To allow end-users to add and remove themselves from a distribution group, you can set the following configuration using the Set-DistributionGroup cmdlet:

Set-DistributionGroup -Identity CompanyNews `

-MemberJoinRestriction Open `

-MemberDepartRestriction Open

This command will allow any user in the organization to join or leave the CompanyNews distribution group without requiring approval by a group owner.

## How it works...

The two parameters that control the membership approval configuration for a distribution group are -MemberJoinRestriction and -MemberDepartRestriction.   
The parameter -MemberJoinRestriction can be set to one of the following values:

* Open: Allows the user to add or remove their account from the group without requiring group owner approval
* Closed: Users cannot join or leave the group.

ApprovalRequired: Requests to join or leave a group must be approved by a group owner

While the parameter -MemberDepartRestriction only can be set to the following values:

* Open: Allows the user to add or remove their account from the group without requiring group owner approval
* Closed: Users cannot join or leave the group.

These settings are not mutually exclusive. For example, you can allow users to join a group without approval, but you can require approval when users try to leave the group, or vice versa. By default, the MemberJoinRestriction property is set to Closed and the MemberDepartRestriction is set to Open.

## There's more...

When member join or depart restrictions are set to ApprovalRequired, a group owner will receive a message informing them of the request, and they can approve or deny the request using the Accept or Reject buttons in Outlook or OWA. The user who created the distribution group will automatically be the owner, but you change the owner, if needed, using the -ManagedBy parameter when running the Set-DistributionGroup cmdlet, as shown:

Set-DistributionGroup -Identity AllEmployees `

-ManagedBy dave@contoso.com,john@contoso.com

As you can see, the -ManagedBy parameter will accept one or more values. If you are setting multiple owners, just separate each one with a comma, as shown previously.

## See also

* Reporting on distribution group membership
* Managing distribution groups in Chapter 3, Managing Recipients

# Creating address lists

Just like dynamic distribution groups, Exchange address lists can be comprised of one or more recipient types and are generated using a recipient filter or using a set of built-in conditions. You can create one or more address list(s), made up of users, contacts, distribution groups, or any other mail-enabled objects in your organization. This recipe will show you how to create an address list using the Exchange Management Shell.

## How to do it...

Let's say we need to create an address list for the sales representatives in our organization. We can use the New-AddressList cmdlet to accomplish this, as shown next:

New-AddressList -Name 'All Sales Users' `

-RecipientContainer contoso.com/Sales `

-IncludedRecipients MailboxUsers

## How it works...

This example uses the New-AddressList cmdlet's built-in conditions to specify the criteria for the recipients that will be included in the list. You can see from the command that, in order for a recipient to be visible in the address list, they must be located within the Sales OU in Active Directory and the recipient type must be MailboxUsers, which only applies to regular mailboxes and does not include other types such as resource mailboxes, distribution groups, and so on.

## There's more...

When you need to create an address list based on a more complex set of conditions, you'll need to use the –RecipientFilter parameter to specify an OPATH filter. For example, the following OPATH filter is not configurable when creating or modifying an address list in EAC/ECP:

New-AddressList -Name MobileUsers `

-RecipientContainer contoso.com `

-RecipientFilter {

HasActiveSyncDevicePartnership -ne $false

}

You can see here that we're creating an address list for all the mobile users in the organization. We've set the RecipientContainer to the root domain, and, within the recipient filter, we've specified that all recipients with an ActiveSync device partnership   
should be included in the list.

A small reminder, update the address list after it’s created so that it applies and updates the list with accurate members. This is done by using the Update-AddressList cmdlet.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | You can create global address lists using the  New-GlobalAddress list cmdlet. |  |

You can combine multiple conditions in your recipient filters using PowerShell's logical operators. For instance, we can extend our previous example to add an additional requirement in the OPATH filter:

New-AddressList -Name MobileUsers `

-RecipientContainer contoso.com `

-RecipientFilter {

(HasActiveSyncDevicePartnership -ne $false) -and

(Phone -ne $null)

}

This time, in addition to having an ActiveSync device partnership, the user must also have a number defined within their Phone attribute in order for them to be included in the list.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | If you need to modify a recipient filter after an address list has already been created, use the Set-AddressList cmdlet. |  |

Exchange supports a various number of both common and advanced properties that can   
be used to construct OPATH filters, as shown in the previous example. To view a list of common filterable properties that can be used with the –RecipientFilter parameter,   
see Appendix A at the end of this book.

## See also

* Working with recipient filters in Chapter 3, Managing Recipients
* Export address list membership to a CSV file

# Exporting address list membership to a CSV file

When it comes to working with address lists, a common task is exporting the list of members to an external file. In this recipe, we'll take a look at the process of exporting the contents of an address list to a CSV file.

## How to do it...

Let's start off with a simple example. The following commands will export the All Users address list to a CSV file:

$allusers = Get-AddressList "All Users"

Get-Recipient -RecipientPreviewFilter $allusers.RecipientFilter |

Select-Object DisplayName,Database |

Export-Csv -Path c:\allusers.csv -NoTypeInformation

When the command completes, a list of user display names and their associated mailbox databases will be exported to c:\allusers.csv.

## How it works...

The first thing we do in this example is create the $allusers variable that stores an   
instance of the All Users address list. We can then run the Get-Recipient cmdlet   
and specify the OPATH filter, using the $allusers.RecipientFilter object as the   
value for the -RecipientPreviewFilter parameter. The results are then piped to the Select-Object cmdlet that grabs the DisplayName and Database properties of the recipient. Finally, the data is exported to a CSV file.

Of course, the given example may not be that practical, as it does not provide the e-mail addresses for the user. We can also export this information, but it requires some special handling on our part. Let's export only the DisplayName and EmailAddresses for each user. To do so, use the following code:

$allusers = Get-AddressList "All Users"

Get-Recipient -RecipientPreviewFilter $allusers.RecipientFilter |

Select-Object DisplayName,

@{n="EmailAddresses";e={$\_.EmailAddresses -join ";"}} |

Export-Csv -Path c:\allusers.csv -NoTypeInformation

Since each recipient can have multiple SMTP e-mail addresses, the EmailAddresses property of each recipient is a multi-valued object. This means we can't simply export this value to an external file, since it is actually an object and not a simple string value. In the given command, we're using the Select-Object cmdlet to create a calculated property for the EmailAddresses collection. Using the -Join operator within the calculated property expression, we are adding each address in the collection to a single string that will be delimited with the semi-colon (;) character.

## There's more...

The given method will work for any of the address lists in your organization. For example, you can export the recipients of the Global Address List (GAL) using the following code:

$GAL = Get-GlobalAddressList "Default Global Address List"

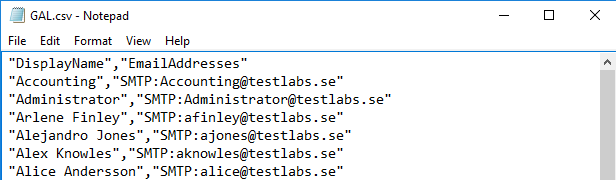
Get-Recipient -RecipientPreviewFilter $GAL.RecipientFilter |

Select-Object DisplayName,

@{n="EmailAddresses";e={$\_.EmailAddresses -join ";"}} |

Export-Csv -Path c:\GAL.csv -NoTypeInformation

As you can see here, the main difference is that this time we are using the   
Get-GlobalAddressList cmdlet to export the default global address list. You can use this technique for any address list in your organization: just specify the name of the address   
list you want to export when using either the Get-AddressList or   
Get-GlobalAddress list cmdlets. The exported csv file will similar to the screenshot below:



7081EN\_05\_08

If you are in a situation where you need to extract all users from a distribution group, as shown above. Besides exporting these into CSV format, they can also be exported into XML format. This is easily done by running the following one-liner:

$dl = Get-DistributionGroupMember -Identity DL\_Marketing

$dl | Export-Clixml -Path C:\Temp\dl.xml

The first row above is used to collect all distribution group members from a group called DL\_Marketing, this by using the cmdlet Get-DistributionGroupMember, the result is saved into the variable $dl. On our next row, we are using the variable to export the distribution group members into an XML file. Use Get-Help Export-Clixml `  
–Detailed to see all the details regarding the cmdlet Export-Clixml.

If we want to have a look at our recent exported content, we can use the cmdlet:

Import-Clixml -Path C:\Temp\dl.xml

## See also

* Exporting reports to text and CSV files in Chapter 2, Exchange Management Shell Common Tasks
* Working with recipient filters in Chapter 3, Managing Recipients
* Creating address lists

# Configuring hierarchical address books

The idea with hierarchical address books is that you can give your users the ability to search for recipients based on your organization's structure, HAB versus the Global Address List which only provides a flat view. HAB was introduced back in Exchange 2010 SP1. The configuration can only be done using the Exchange Management Shell, and, in this recipe, we'll take a look at an example of how you can configure this feature in your organization. If you’re planning on using Outlook with Exchange 2016, it requires at least the Outlook version 2010 with KB2965295 or newer.

## How to do it...

1. It is recommended that you create an OU in Active Directory to store the root HAB objects. You can create a new OU using your Active Directory administrations tools, or using PowerShell. The following code can be used to create an OU in the root of the Contoso domain called HAB:

$objDomain = [ADSI]''

$objOU = $objDomain.Create('organizationalUnit', 'ou=HAB')

$objOU.SetInfo()

1. Next, create a root distribution group for the HAB:

New-DistributionGroup -Name ContosoRoot `

-DisplayName ContosoRoot `

-Alias ContosoRoot `

-OrganizationalUnit contoso.com/HAB `

-SamAccountName ContosoRoot `

-Type Distribution `  
-IgnoreNamingPolicy

1. Configure the Contoso distribution group as the root organization for the HAB:

Set-OrganizationConfig -HierarchicalAddressBookRoot ContosoRoot

1. At this point, you need to add subordinate groups to the root organization group. These can be existing groups or you can create new ones. In this example, we'll add three existing groups called Executives, Finance and Sales to the root organization in the HAB:

Add-DistributionGroupMember -Identity ContosoRoot -Member Executives

Add-DistributionGroupMember -Identity ContosoRoot -Member Finance

Add-DistributionGroupMember -Identity ContosoRoot -Member Sales

1. Finally, we'll designate each of the groups as hierarchical groups and set the seniority index for the subordinate groups:

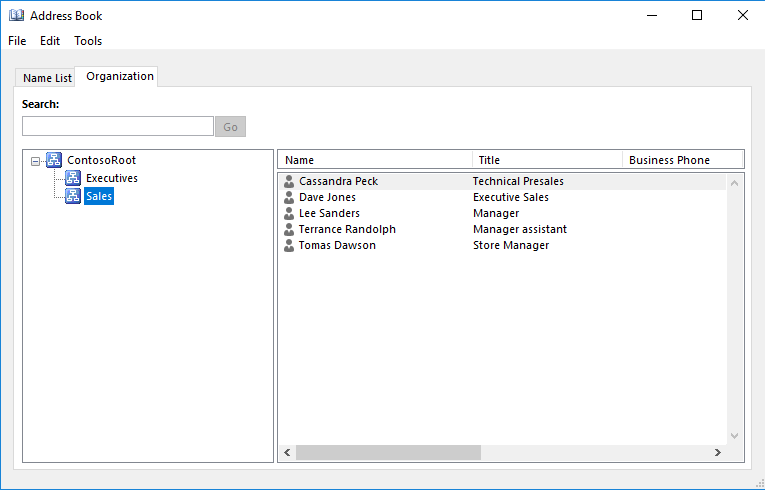
Set-Group -Identity ContosoRoot -IsHierarchicalGroup $true

Set-Group Executives -IsHierarchicalGroup $true -SeniorityIndex 100

Set-Group Finance -IsHierarchicalGroup $true -SeniorityIndex 50

Set-Group Sales -IsHierarchicalGroup $true -SeniorityIndex 75

1. After this configuration has been completed, Outlook 2010+ users can click on the Address Book button and view a new tab called Organization that will list our HAB:



7081EN\_05\_09

## How it works...

The root organization of a HAB is used as the top tier for the organization. Under the root, you can add multiple tiers by adding other distribution groups as members of this root tier and configuring them as hierarchical groups. This allows you to create a HAB that is organized by department, location, or any other structure that makes sense for your environment.

In order to control the structure of the HAB, you can set the SeniorityIndex of each sub group under the root organization. This index overrides the automatic sort order based on the DisplayName which would otherwise be used if no value was defined. This also works for individual recipients within each group. For example, you can set the SeniorityIndex on each member of the Executives group using the Set-User cmdlet:

Set-User cmunoz -SeniorityIndex 100

Set-User awood -SeniorityIndex 90

Set-User ahunter -SeniorityIndex 80

The users will be displayed in order, with the highest index number first. This allows you to further organize the HAB and override the default sort order if needed.

## There's more...

You may notice that after configuring a HAB that Outlook 2010+ users are not seeing the Organization tab when viewing the Address Book. If this happens, double check the Active Directory schema attribute ms-Exch-HAB-Root-Department-Link using ADSIEdit. The isMemberOfPartialAttributeSet attribute should be set to True. If it is not, change this attribute to True, ensure that this has replicated to all DCs in the forest, and restart the Microsoft Exchange Active Directory Topology service on each Exchange server. Of course, this is something you'll want to do during out of hours to ensure there is no disruption of service for end-users. After this change has been completed, Outlook 2010+ users should be able to view the Organization tab in the Address Book.

We can make a basic script for the tasks above, save it to a file called to hab.ps1:

$objDomain = [ADSI]''

$objOU = $objDomain.Create('organizationalUnit', 'ou=HAB')

$objOU.SetInfo()

$hab = Read-Host "Type a name for the HAB root"

New-DistributionGroup -Name $hab -DisplayName $hab `  
-OrganizationalUnit "contoso.com/HAB" -SAMAccountName $hab `  
-Type Distribution

Set-Group -Identity $hab –IsHierarchicalGroup $true

$File = Import-Csv ".\hab.csv"

$File | foreach {

New-DistributionGroup -Name $\_.GroupName -OrganizationalUnit `  
 "contoso.com/HAB" -Type Distribution -SAMAccountName $\_.GroupName

Set-Group -Identity $\_.GroupName -SeniorityIndex $\_.Seniority `  
 –IsHierarchicalGroup $true

Add-DistributionGroupMember -Identity $hab -Member $\_.GroupName

}

Set-OrganizationConfig -HierarchicalAddressBookRoot $hab

The script starts with creating an OU called HAB. A variable is created with the name $hab, the prompt will ask for a name for the HAB root. This name is used when creating the initial distribution group. A CSV file called hab.csv is imported to the variable $File, a foreach loop is used for processing each row in the CSV file.

The CSV includes names of the distribution groups and what SeniorityIndex value they should be configured with. The script also makes sure that these groups are configured as hierarchical groups. Finally, the users needs to be added into the corresponding groups for forming the organization.

## See also

Managing distribution groups in Chapter 3, Managing Recipients