Chapter 6: Managing Mailboxes

Gareth Gudger

In this chapter, we look at mailboxes. You will learn about the different types of mailboxes such as:

- User Mailboxes
- Office 365 Mailboxes (Remote Mailboxes)
- Archive Mailboxes
- Shared Mailboxes
- Room Mailboxes
- Equipment Mailboxes
- Other mailbox types (Linked, Site, Arbitration & Discovery Mailboxes)

In addition, we discuss how to perform common administrative tasks through both the Exchange Admin Center (EAC) and the Exchange Management Shell (EMS). Tasks that include:

- Creating, enabling, deleting and purging mailboxes
- Managing mailbox properties and features
- Managing mailbox folder permissions and delegations
- Bulk creation and manipulation of mailboxes
- Moving mailboxes
- Mailbox usage and move reporting

User Mailboxes

A user mailbox is the most common type of mailbox in any organization. A user mailbox is assigned to an individual and allows them to send, receive and store email messages. In addition, they can use their mailbox for calendaring, organizing meetings, managing contacts, creating tasks and, storing other types of data. Mailboxes are stored in databases which in turn are hosted by an Exchange server. Certain properties such as the users email address is stored in Active Directory as an attribute on their user account. A user can access their mailbox through a myriad of solutions including Outlook, Outlook on the Web, or, through a mobile device. In the following sections, we will look at how to create, manage and delete user mailboxes.

Mailbox-enabling an existing user

If you previously created a user in Active Directory (or, you are operating in a split-permissions model) you may have users that need to be assigned mailboxes. This is referred to mail-enabling a user. First let's examine how to do this in the Exchange Admin Center (EAC).

In this example, we are going to mail-enable user Brian Best. Brian already has an account in Active Directory but has no mailbox. Brian's email will be brian.best@space-corp.net.

Log into the Exchange Admin Center (EAC) and select the **Recipients** tab on the left and **Mailboxes** sub tab across the top. From here click the **New** button (represented by a plus sign) and select **User Mailbox**.

On the *New User Mailbox* window (Figure 6-1) type an alias for the user. In our example we have typed *brian.best*.

Note: The alias must be unique across the organization. An alias can be up to 64 characters in length and contain letters, numbers and the characters # $ % & ' * + - / = ^ _ ` { } | ~ ! ? and periods (.). The default

Email Address Policy uses the alias to create the users email address. For more information on Email Address Policies refer to *Chapter 9: Managing Addressing*.

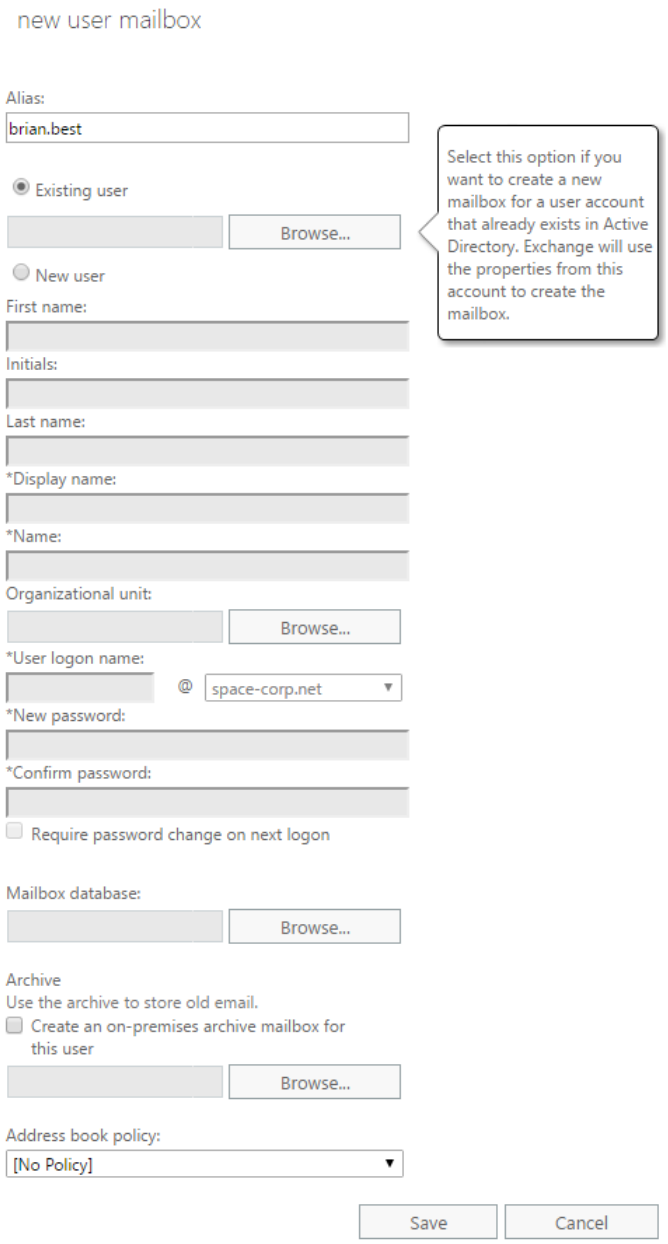

Figure 0-1: Mail-enabling an existing user

Select **Existing User** and click the **Browse** button.

From the *Select User* dialog pick the user you wish to mail enable. You can also search for the user on this page to filter the choices. Click **Ok**.

At this point, Exchange has all the information it needs to mail-enable our user Brian. On the *New User Mailbox* window, we could simply click the **Save** button and our task would be complete. First let's explore **More Options** link at the bottom of the page. Clicking this link presents us with a couple extra settings.

The first is *Mailbox Database*. If we click **Browse**, we can select which mailbox database we want our user Brian to be created on. If we leave this field blank Exchange will automatically choose the best database for us.

Note: For more information on automatic mailbox distribution check this article. https://technet.microsoft.com/en-us/library/ff477621(v=exchg.150).aspx

The next option is *Archive*. By selecting the checkbox **Create an on-premises archive mailbox for this user** we have the option of creating an archive at the same time the primary mailbox is created. The **Browse** button allows us to specify which database hosts the user's archive. This can be any database, including the one that hosts the user's primary mailbox. An archive mailbox can be added at any time so it isn't crucial to do this now. We will cover archive mailboxes later in this chapter.

The final option is to specify an *Address Book Policy*. An Address Book Policy is a way to give a user a custom address book. An Address Book Policy can be incredibly useful when you need to segment the user population. Examples of where this may be necessary could be the result of an acquisition, divestiture, or, a multi-tenant Exchange environment. More information of Address Book Policies can be found in *Chapter 9: Managing Addresses*.

For our example, we will leave all the additional options as blank. With just the *alias* and *existing user* fields populated click the **Save** button. Your mail-enabled user will now appear under the **Mailboxes** tab.

Now let's look at how to mail-enable a user in the Exchange Management Shell (EMS). To do this we will use the **Enable-Mailbox** cmdlet. Using the above example of Brian Best let's see what the process would have looked like in PowerShell.

```
[PS] C:\> Enable-Mailbox -Identity "Brian Best" -Alias "brian.best"
```

In this example, **-Identity** specifies the user to mail enable is Brian Best and that his unique **-Alias** should be brian.best.

If we wanted to specify which database Brian should be assigned, we would need to add the **-Database** parameter. In the example below we add Brian to database DB01.

```
[PS] C:\> Enable-Mailbox -Identity "Brian Best" -Alias "brian.best" -Database "DB01"
```

If we want to specify the creation of an archive in addition to the primary mailbox we would need to first mail-enable Brian with one of the preceding commands. Then we would run a second instance of the **Enable-Mailbox** cmdlet with the **-Archive** parameter. In our example, we will also specify that the archive should be hosted in a different database called DB02. We will use the **-ArchiveDatabase** parameter for this purpose.

```
[PS] C:\> Enable-Mailbox -Identity "Brian Best" -Archive -ArchiveDatabase "DB02"
```

Note: For an extensive list of all available parameters for **Enable-Mailbox** cmdlet check the following article https://technet.microsoft.com/en-us/library/aa998251(v=exchg.160).aspx

Creating a new user and mailbox

In our last section, we discussed how to mailbox-enable an existing user. In this section, we will look at how to create a new user and a new mailbox in a single workflow. We will explore how to accomplish this task with both the EAC and EMS.

In this example, we are going to create a new user called Sarah Gibbs. Sarah does not currently have a user account in Active Directory. Sarah's email will be sarah.gibbs@space-corp.net.

Log into the Exchange Admin Center (EAC) and select the **Recipients** tab on the left and **Mailboxes** sub tab across the top. From here click the **New** button (represented by a plus sign) and select **User Mailbox**.

On the *New User Mailbox* window (Figure 6-2) type an alias for the user. In our example we have typed *sarah.gibbs*. Select **New User** and provide information about your user, such as first name, last name, display name, username and password.

Filling in the **First name** and **Last name** fields will automatically populate the **Display Name** and **Name** fields. These fields can be modified after they have been populated.

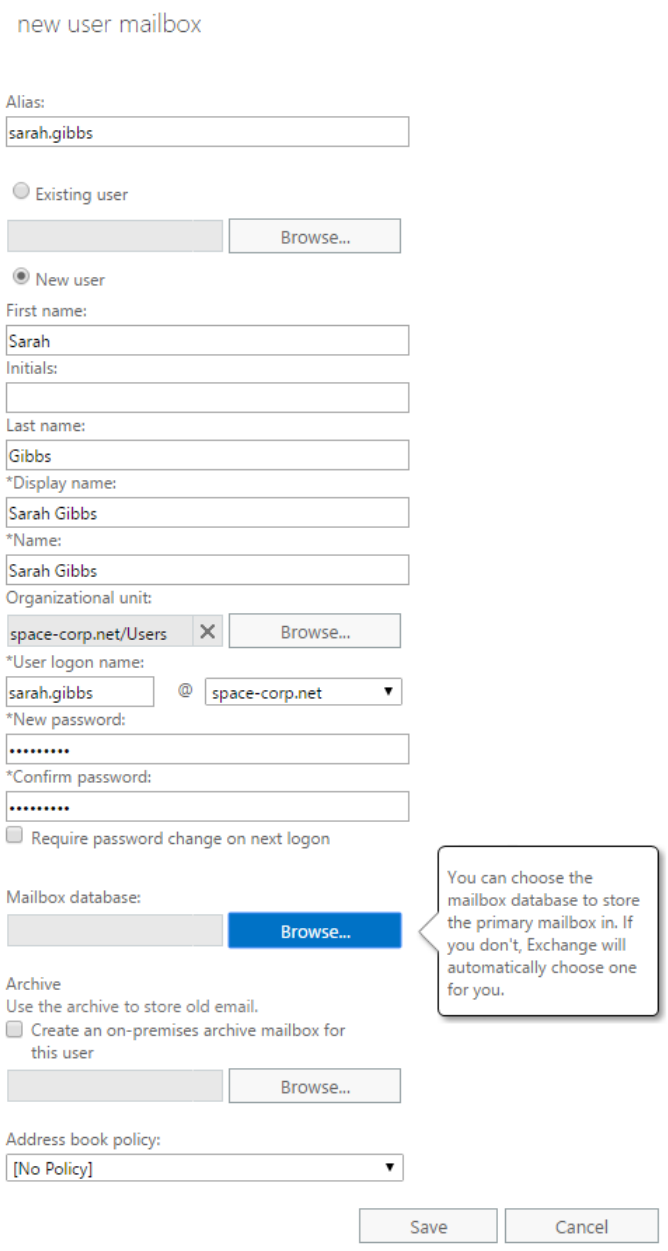

Figure 0-2: Creating a new user and mailbox

The Display Name is what other users will see when they search for that user in the address book. Based on our example in Figure 6-2, other users will see "Sarah Gibbs" in their address book. This is the same name that will be seen on all *To:* and *From:* lines in Outlook or Outlook on the Web.

The Name field corresponds to the Display Name field in Active Directory. Display Name and Name can hold different values.

Click **Browse** in the Organizational Unit section. This brings up the *Select an Organization Unit* dialog. From here you can select which Organization Unit (OU) you want the new user account to be created under. You can also search for an OU to filter the results. It is worth noting that keeping the Organization Unit field blank will result in the user being created in the built-in Users OU under the root of your domain.

Under **User logon name** specify a new user name for the user. From the drop-down to the right of the @ symbol pick the domain suffix for the user. This builds a User Principal Name (UPN) for the user which they can use to log onto domain resources. These fields match those when creating a new user in Active Directory Users and Computers.

Specify and confirm a password in the **New Password** and **Confirm Password** fields. You can also check the box to **Require password change on next logon** to force the user to create a new password.

If you are looking to create a user and mailbox quickly the minimum fields required are those marked with an asterisk.

We have discussed the *More Options* link in the previous section. These remain the same regardless of whether you pick existing or new user.

For our example, we specified an alias, first name, last name (which populated the display name and name fields for us), Organizational Unit, username, domain suffix and password.

With your fields populated click the **Save** button. Your mailbox-enabled user will now appear under the **Mailboxes** tab.

Let's explore how we would have completed the same task but using the Exchange Management Shell (EMS). To do this we will use the **New-Mailbox** cmdlet. Using the above example of Sarah Gibbs let's see what the process would have looked like in PowerShell.

First, we will need to capture a temporary password for the user in a variable. The password will be saved as a secure string. To do this enter the following command. Enter a password when prompted.

```
[PS] C:\> $password = Read-Host "Enter password" -AsSecureString
Enter Password: *********
[PS] C:\>
```

Next, let's create the mailbox and parse in the $password variable.

```
[PS] C:\> New-Mailbox -Name "Sarah Gibbs" -DisplayName "Sarah Gibbs" -FirstName "Sarah" -LastName
"Gibbs" -UserPrincipalName "sarah.gibbs@space-corp.net" -Password $password -Alias "sarah.gibbs" -
OrganizationalUnit "space-corp.net/Users" -ResetPasswordOnNextLogon:$true
```

In this command:

-Name specifies the content of the display name field in Active Directory.

-DisplayName specifies the name seen in Exchange, such as in the Address Book, To and From lines.

-FirstName specifies the first name of the user.

-LastName specifies the last name of the user.

-UserPrincipalName specifies the user name in UPN form.

-Password calls the variable named *$password* from our previous command.

-Alias specifies a unique mail nickname and forms the initial email address based on the default Email Address Policy (EAP).

-OrganizationalUnit specifies where in Active Directory the new user account should be created. In our example, we specified this as the Users OU. This is an optional parameter. Leaving this parameter out will place the user in the Users OU in the root of the domain.

-ResetPasswordOnNextLogon specifies whether a user must change their password the next time they log in. This is a Boolean response of either $true or $false. In our example, we set this to true forcing the user to change their password at next log on.

After running this cmdlet Exchange will create new user, Sarah Gibbs, in the Users OU in the root of Space-Corp.net and allow Exchange to determine which database to create her new mailbox.

Like how we did when we mailbox-enabled an existing user, we can also specify a database for the user with the **-Database** parameter. In this example, we specify that Sarah Gibbs should be created in database DB01.

```
[PS] C:\> New-Mailbox -Name "Sarah Gibbs" -DisplayName "Sarah Gibbs" -FirstName "Sarah" -LastName
"Gibbs" -UserPrincipalName "sarah.gibbs@space-corp.net" -Password $password -Alias "sarah.gibbs" -
OrganizationalUnit "space-corp.net/Users" -Database "DB01" -ResetPasswordOnNextLogon:$true
```

If we wanted to create an archive along with her primary mailbox the command would look like this. In addition, we are also specifying that we want her archive mailbox in a separate database called DB02.

```
[PS] C:\> New-Mailbox -Name "Sarah Gibbs" -DisplayName "Sarah Gibbs" -FirstName "Sarah" -LastName
"Gibbs" -UserPrincipalName "sarah.gibbs@space-corp.net" -Password $password -Alias "sarah.gibbs" -
OrganizationalUnit "space-corp.net/Users" -Database "DB01" -ResetPasswordOnNextLogon:$true -
Archive:$true -ArchiveDatabase "DB02"
```

It is also possible to specify a password without using a variable and thus combining it into a single command. The drawback here is that the password would be visible on the screen in plain text so is considered less secure. The prior method obfuscates the password by changing it to asterisks as it is typed. Here is the less secure version of the command.

```
[PS] C:\> New-Mailbox -Name "Sarah Gibbs" -DisplayName "Sarah Gibbs" -FirstName "Sarah" -LastName
"Gibbs" -UserPrincipalName "sarah.gibbs@space-corp.net" -Password (ConvertTo-SecureString -String
P@ssw0rd -AsPlainText -Force) -Alias "sarah.gibbs" -OrganizationalUnit "space-corp.net/Users" -
ResetPasswordOnNextLogon:$true
```

Unlike its predecessor the command above contains the password (of *P@ssw0rd)* in cleartext. The method you choose will in large be governed by your security policy.

Note: For an extensive list of all available parameters for **New-Mailbox** cmdlet check the following article https://technet.microsoft.com/en-us/library/aa997663(v=exchg.160).aspx

Creating an Office 365 (remote) mailbox

If you are operating Exchange in hybrid with Office 365, then you will see one additional option when pressing the **New** (plus sign) button under the **Mailboxes** tab. This option is only available if you have established hybrid mode with Office 365. The option is **Office 365 mailbox** (Figure 6-3).

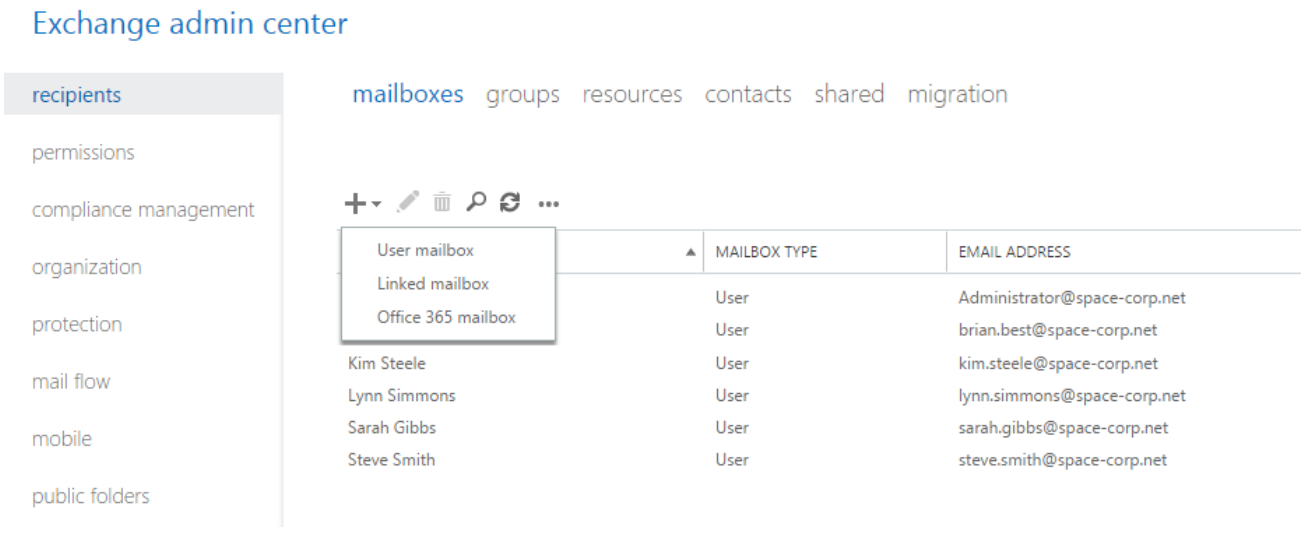

Figure 0-3: New Office 365 Mailbox

Selecting this option walks you through the process of creating a remote mailbox in Office 365. The benefit here is that you do not need to migrate the mailbox after it is created as it already exists in the cloud. Keep in mind that you will not see this mailbox in the Office 365 tenant until directory synchronization has run.

We will look at how to create an Office 365 mailbox in the EAC first.

In this example, we are going to create a new user called Fred Schmidt who will have a mailbox in Office 365. Fred does not currently have a user account in Active Directory. Fred's email will be fred.schmidt@space-corp.net.

Select the **Recipients** tab on the left and **Mailboxes** sub tab across the top. From here click the **New** button (represented by a plus sign) and select **Office 365 Mailbox**.

On the *New Office 365 Mailbox* window (Figure 6-4) type the **First name** and **Last name** of the user. As you complete these fields you will notice that the **Name** field populates combining these values. The name field corresponds to the display name field for the user object in Active Directory. You can alter this field to be a different value than what was suggested.

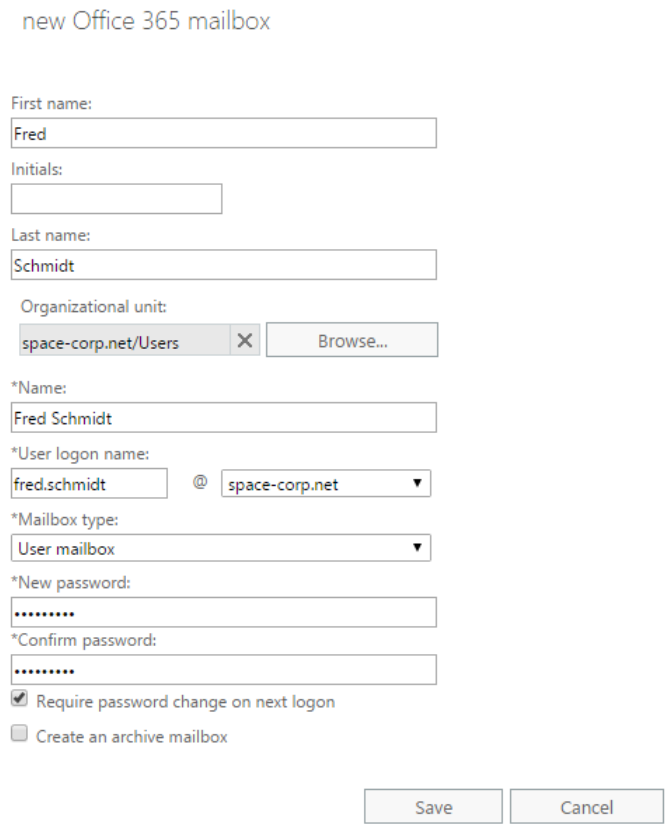

Figure 0-4: New Office 365 Mailbox Dialog

Click **Browse** in the Organizational Unit section. This brings up the *Select an Organization Unit* dialog. From here you can select which Organization Unit (OU) you want the new user account to be created under.

Under **User logon name** specify a new user name for the user. From the drop-down to the right of the @ symbol pick the domain suffix for the user. This builds a User Principal Name (UPN) for the user.

Under *Mailbox type* pick the type of mailbox you want to create in Office 365. In our example, we are creating a mailbox for user Fred Schmidt so we will pick **User Mailbox**. Room and Equipment mailboxes are also available. We will cover those types of mailboxes later in this chapter.

Specify and confirm a password in the **New Password** and **Confirm Password** fields. You can also check the box to **Require password change on next logon** to force the user to create a new password.

Be selecting **Create an archive mailbox** we can also instruct Office 365 to create an archive mailbox for the user in the cloud.

Unlike on-premises mailbox creation we do not get an option to pick a primary or archive database for our user. In Office 365 we cannot manage databases or servers. In turn this negates our ability to choose how those users are assigned or distributed across databases. Microsoft makes this choice for us and it is not uncommon for Microsoft to redistribute mailboxes across databases.

If you are looking to create a user and mailbox quickly the minimum fields required are those marked with an asterisk.

For our example, we specified a first name, last name (which populated name field for us), Organizational Unit, username, domain suffix, mailbox type and password.

With your fields populated click the **Save** button. Your mail-enabled user will now appear under the **Mailboxes** tab. Unlike the mailboxes we created in the previous sections there is one subtle difference. The *Mailbox Type* column will list your user as Office 365. On-premises mailboxes will be listed as User (Figure 6-5).

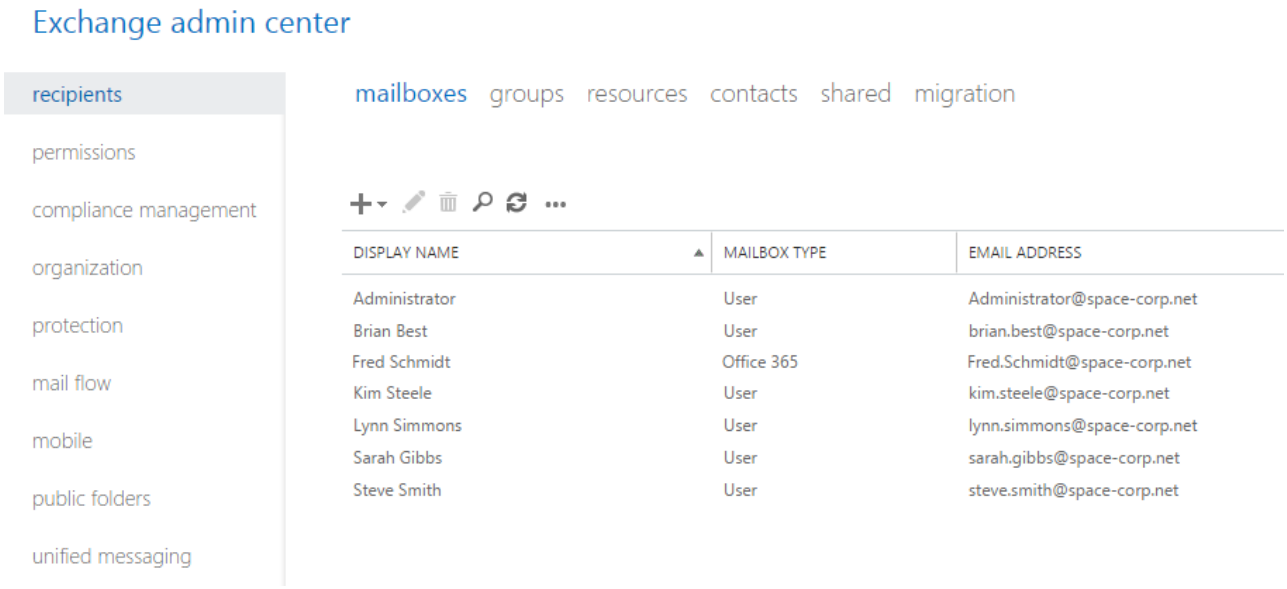

Figure 0-5: User vs. Office 365 Mailbox Type

Let's explore how we would have completed the same task but using EMS. To do this we will use the **New-RemoteMailbox** cmdlet. Using the above example of Fred Schmidt let's see what the process would have looked like in PowerShell.

First, we will need to capture a temporary password for the user in a variable. The password will be saved as a secure string. To do this enter the following command. Enter a password when prompted.

```
[PS] C:\> $password = Read-Host "Enter password" -AsSecureString
Enter Password: *********
[PS] C:\>
```

Next, let's create the mailbox and parse in the $password variable.

```
[PS] C:\> New-RemoteMailbox -Name "Fred Schmidt" -FirstName "Fred" -LastName "Schmidt" -
OnPremisesOrganizationalUnit "space-corp.net/Users" -UserPrincipalName "fred.schmidt@space-corp.net"
-Password $password -ResetPasswordOnNextLogon:$true
```

In this cmdlet:

-Name specifies the content of the display name field in Active Directory.

-FirstName specifies the first name of the user.

-LastName specifies the last name of the user.

-OnPremisesOrganizationalUnit specifies where in the on-premises Active Directory the new user account should be created. The New-RemoteMailbox cmdlet tweaks the naming of the parameter from -OrganizationalUnit to -OnPremisesOrganizationUnit to emphasize where the user account exists. In our example, we specified this as the on-premises Users OU in the space-corp.net domain. This is an optional parameter.

-UserPrincipalName specifies the user name in UPN form.

-Password calls the variable named *$password* from our previous command.

-ResetPasswordOnNextLogon specifies whether a user must change their password the next time they log in.

After running this command Exchange will create new user, Fred Schmidt, in the Users OU in the root of Space-Corp.net with a remote mailbox on Office 365. Once directory synchronization performs a full or delta sync the mailbox will be present in the service. Keep in mind the user will need to be assigned an Office 365 license before they can access their mailbox.

More information of implementing hybrid with Office 365 can be found in *Chapter 13: Updates and Migration*.

Note: For an extensive list of all available parameters for **New-RemoteMailbox** cmdlet check the following article https://technet.microsoft.com/en-us/library/ff607480(v=exchg.160).aspx

Adding users in bulk

Adding users one at a time through either the EAC or EMS can be time consuming and a bit mind-numbing. The benefit of EMS is that you can get creative when mass creating users. In fact, some administrators have gotten so creative they automate the creation of users and mailboxes by pulling new employee information directly out of another system, such as a Human Resources database. In this section, we will look at bulk creating users and mailboxes through data contained in a Comma Separated Value (CSV) file.

Not all the columns identified in Figure 6-6 are required. Some are optional. Going back to the earlier section titled '*Creating a new user and mailbox*' we had discussed how only the fields marked with an asterisk were required. Using Figure 6-2 as an example we can see that First Name and Last Name are two columns we could drop from our CSV. On the flipside, there are other columns we could add. If we wanted to control where mailboxes were created, we could add a database column to this file.

	A	B	C	D	E	F	G
1	FirstName	LastName	Name	DisplayName	OrganizationalUnit	UserPrincipleName	
2	Pam	Kipling	Pam Kipling	Pam Kipling	space-corp.net/Employees/R&D	pam.kipling@space-corp.net	
3	Alex	Short	Alex Short	Alex Short	space-corp.net/Employees/R&D	alex.short@space-corp.net	
4	Rachel	Hughes	Rachel Hughes	Rachel Hughes	space-corp.net/Employees/R&D	rachel.hughes@space-corp.net	
5	Kendal	Ayers	Kendal Ayers	Kendal Ayers	space-corp.net/Employees/Production	kendal.ayers@space-corp.net	
6	Nick	Haywood	Nick Haywood	Nick Haywood	space-corp.net/Employees/Production	nick.haywood@space-corp.net	
7	Ed	Patterson	Ed Patterson	Ed Patterson	space-corp.net/Employees/Production	ed.patterson@space-corp.net	
8	Joe	Diaz	Joe Diaz	Joe Diaz	space-corp.net/Employees/Sales	joe.diaz@space-corp.net	
9	Fred	Schmidt	Fred Schmidt	Fred Schmidt	space-corp.net/Employees/Sales	fred.schmidt@space-corp.net	
10	Morgan	Clarkson	Morgan Clarkson	Morgan Clarkson	space-corp.net/Employees/Corporate	morgan.clarkson@space-corp.net	
11	Harvey	Lovell	Harvey Lovell	Harvey Lovell	space-corp.net/Employees/Corporate	harvey.lovell@space-corp.net	
12							
13							
14							

Figure 0-6: Sample CSV User Import File

It is worth noting that the column headers can be any name you desire. However, for ease of troubleshooting we recommend keeping these columns named after the parameters they serve. In the next series of commands, we show how to match each parameter with its corresponding column header. Let's get started.

Our first task is to populate a variable with an initial password. In our example, we will make this *Welcome123*. This password will be set across all users we create. We will be sure to force the users to change their password at logon in the subsequent command. To set an initial password issue the following command.

```
[PS] C:\> $Password = Read-Host "Enter Password" -AsSecureString
Enter Password: *********
[PS] C:\>
```

Next, we will import our CSV and pipe it into a **ForEach** loop. To do this we use the **Import-CSV** command and specify the location of the file with the **–Path** parameter. The ForEach loop will then run a single instance of the **New-Mailbox** command against each line in our CSV file.

```
[PS] C:\> Import-CSV -Path "C:\EmployeeData\NewEmployeeInfo.csv" | ForEach {New-Mailbox -FirstName
$_.FirstName -LastName $_.LastName -Name $_.Name -DisplayName $_.DisplayName -UserPrincipalName
$_.UserPrincipleName -OrganizationalUnit $_."OrganizationalUnit" -Password $Password -
ResetPasswordOnNextLogon $true}

Name                  Alias                 ServerName       ProhibitSendQuota
----                  -----                 ----------       -----------------
Pam Kipling           pam.kipling           exch01           Unlimited
Alex Short            alex.short            exch01           Unlimited
Rachel Hughes         rachel.hughes         exch01           Unlimited
Kendal Ayers          kendal.ayers          exch02           Unlimited
Nick Haywood          nick.haywood          exch01           Unlimited
Ed Patterson          ed.patterson          exch01           Unlimited
Joe Diaz              joe.diaz              exch02           Unlimited
Fred Schmidt          fred.schmidt          exch02           Unlimited
Morgan Clarkson       morgan.clarkson       exch02           Unlimited
Harvey Lovell         harvey.lovell         exch01           Unlimited
```

In the New-Mailbox command we match each parameter with a column in the CSV file. For example, we match the **–FirstName** parameter to the column titled FirstName from our CSV. We use the dollar-underscore-period " $_. " to specify that the values should be pulled from the CSV file that is currently stored in the pipe. The **–Password** parameter retrieves the value we set in $Password variable. The **–ResetPasswordOnNextLogon** forces the user to change their password once they log in.

If you find that all user accounts are created with missing information that was present in your CSV file, make sure the column specified in the New-Mailbox command and the column in the CSV file match. A mismatch will cause the script to skip that column entirely.

If you find that the user accounts are being created in the Users OU versus the OU specified, then the script is unable to find that OU. Note that the OU's must be created beforehand. When an OU cannot be found, it is the same as if you omitted the **–OrganizationalUnit** parameter altogether. The absence of this parameter places all user accounts under the default Users OU in the root of your domain.

It is also possible to include a password column in our CSV file. The danger here is that our passwords are stored in clear text. We could store our initial password of *Welcome123* this way and combine our two commands into the following:

```
Import-CSV -Path "C:\EmployeeData\NewEmployeeInfo.csv" | ForEach {New-Mailbox -FirstName
$_.FirstName -LastName $_.LastName -Name $_.Name -DisplayName $_.DisplayName -UserPrincipalName
$_.UserPrincipleName -OrganizationalUnit $_."OrganizationalUnit" -Password (ConvertTo-SecureString
$_.Password -AsPlainText -Force) -ResetPasswordOnNextLogon $true}
```

Disabling a mailbox

Like how we can mailbox-enable a user we can also mailbox-disable a user. Disabling a mailbox involves disconnecting the mailbox from the user and removing any mail attributes the user may have had. The user account is saved while the mailbox enters a disconnected state. By default, Exchange retains disconnected mailboxes for up to 30 days. The benefit here is that the mailbox can be reconnected during this period to the

same user account, or, a different user account. If the mailbox is not reconnected before the 30 days has passed, then it is deleted. The user account remains unaffected. A mailbox can also be purged before the 30 days is up.

Let's explore the process of disabling a mailbox in the EAC first.

In this example, we are going to disable Sarah Gibbs' mailbox.

From the **Recipients** tab on the left select the **Mailboxes** sub tab across the top. From here select the user you wish to mailbox-disable. Click the **More** button (represented by an ellipsis) and select **Disable** (Figure 6-7).

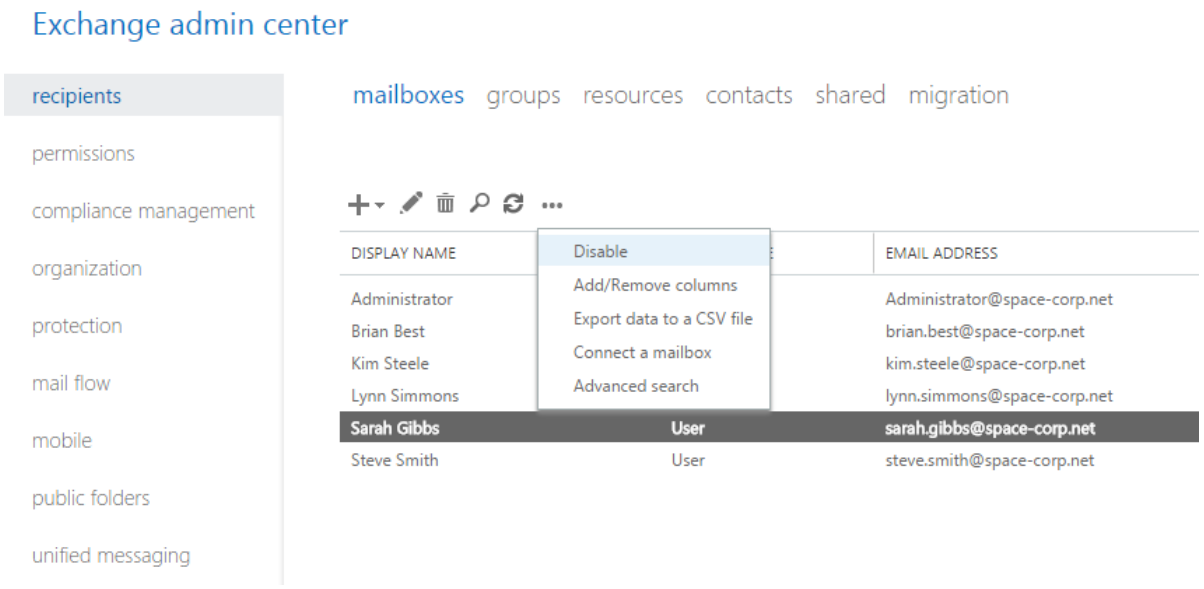

Figure 0-7: Disabling a user mailbox

EAC will present a warning asking us to confirm the request. Click **Yes** to disable your user. The user will disappear from the mailboxes view.

Let's explore how we would have completed the same task but using the EMS. To do this we will use the **Disable-Mailbox** cmdlet. Using the above example of Sarah Gibbs let's see what the process would have looked like in PowerShell.

```
[PS] C:\> Disable-Mailbox -Identity "Sarah Gibbs"

Confirm
Are you sure you want to perform this action?
Disabling mailbox "Sarah Gibbs" will remove the Exchange properties from the Active Directory user
object and mark the mailbox in the database for removal. If the mailbox has an archive or remote
archive, the archive will also be marked for removal. In the case of remote archives, this action is
permanent. You can't reconnect this user to the remote archive again.

[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [?] Help (default is "Y"):
```

In the command above the **-Identity** parameter specifies that the mailbox for Sarah Gibbs should be disconnected from her user account. We are prompted to confirm whether we want this action to proceed. Hitting enter will take the default action of Yes. The mailbox is then retained in a disconnected state as per the value of the mailbox retention period under the properties of the database. As mentioned earlier the default value for this is 30 days. Sarah's Active Directory account is also stripped of its mail attributes but the account itself is maintained.

Note: For an extensive list of all available parameters for **Disable-Mailbox** cmdlet check the following article https://technet.microsoft.com/en-us/library/aa997210(v=exchg.160).aspx

Disabling a mailbox and deleting its user

Unlike its disable counterpart the delete button takes the process one step further. Not only does the delete button put the mailbox in a disconnected state but it also removes the user account from Active Directory.

Warning: It is unfortunate that the delete button appears on the main toolbar and the disable button is tucked away under the ellipsis menu as the delete button is much more destructive. It is not uncommon for the delete and disable functions to be confused so it is important to reiterate that disable maintains the user account whereas delete button removes the user account entirely. To recover the user account, you would need to leverage the Active Directory Recycle Bin or another form of object recovery.

Let's look at how to disconnect a mailbox and delete the user. For this example, we will delete user Brian Best. Let's follow this process first in the EAC.

Select the **Recipients** tab on the left and **Mailboxes** sub tab across the top. From here select the user you wish to delete. Click the **Delete** button (represented by a garbage can). You will be prompted to confirm the deletion of your user. Click **Yes**.

In our example, Brian's mailbox is put into a disconnected state and his user account is deleted from Active Directory. We can reconnect Brian's mailbox to another user account, who is not mailbox-enabled, as long as the mailbox retention period has not expired.

Let's see how we would have completed the same task but using the EMS. To do this we will use the **Remove-Mailbox** cmdlet.

```
[PS] C:\> Remove-Mailbox -Identity "Brian Best"

Confirm
Are you sure you want to perform this action?
Removing mailbox "Brian Best" will remove the Active Directory user object and mark the mailbox and
the archive (if
present) in the database for removal.
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [?] Help (default is "Y"):
```

We are prompted to confirm whether we want this action to proceed. Pressing **Enter** will take the default action of Yes. The mailbox is retained in a disconnected state as per the value of the mailbox retention period. However, Brian's Active Directory account is immediately deleted.

Reconnecting a disabled mailbox

If you have disabled a mailbox using the methods described in the last two sections you can reconnect that mailbox, provided the mailbox retention period has not been exceeded. By default, that mailbox retention period is 30 days.

Let's explore the process of reconnecting a disconnected mailbox. We will explore this process in the EAC first. In this example, we are going to reconnect Sarah Gibbs' mailbox to her Active Directory user account.

From the **Recipients** tab on the left select **Mailboxes** sub tab across the top. From here select the user you wish to mailbox-enable. Click the **More** button (represented by an ellipsis) and select **Connect a Mailbox**.

From the *Connect a mailbox* dialog select the user you wish to reconnect and click the **Connect** button (Figure 6-8). In our example, we have selected the mailbox with the display name of Sarah Gibbs.

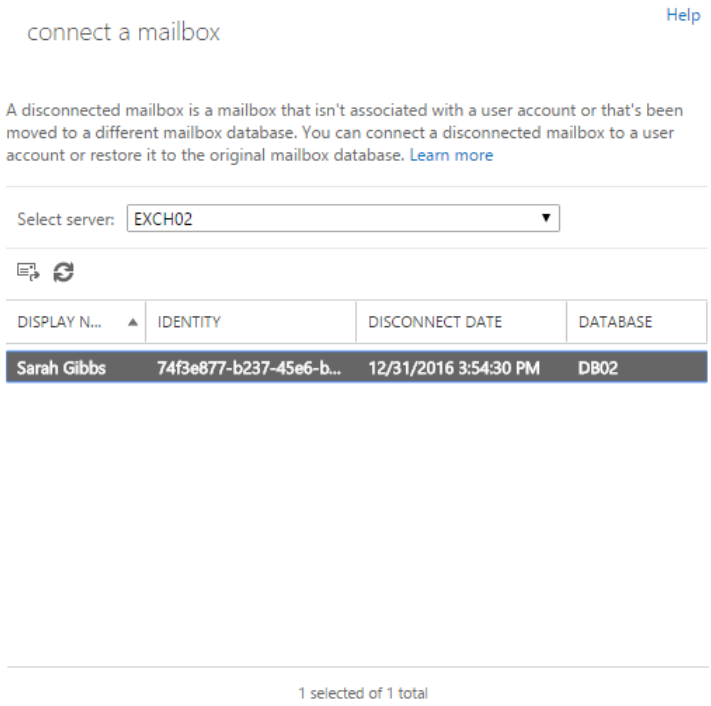

Figure 0-8: Connecting a disconnecting mailbox

An *Information* dialog will appear asking if you want to reattach the mailbox to the original user, or, a different user (Figure 6-9). To connect to the original user, select **Yes, connect to the user account above**. To connect the mailbox to a different user, select **No, I want to connect to a different user account**.

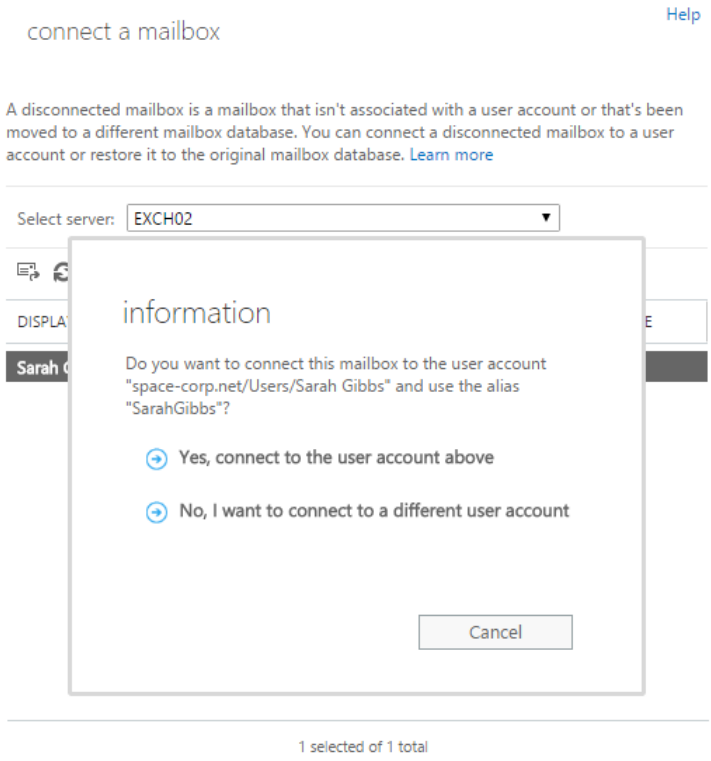

Figure 0-9: Connecting a disconnected mailbox to the same user

Real World: If you are testing this function in a lab and your disconnected mailbox is not visible then it may be a result of an uninitialized mailbox. For a disconnected mailbox to be available for reconnection it must have been logged into at least once by the user.

If you reconnect the mailbox back to the original user account, the process will take a couple of seconds and put you back to the mailboxes tab. Once you hit the **Refresh** button the user and their mailbox will reappear under the mailboxes tab.

If we choose to reconnect to another user, the wizard continues. First it asks whether you want to alter the type of mailbox during reconnection. For example, what was originally a user mailbox could be transformed to a shared mailbox. Once you have confirmed the type of mailbox click **Next**.

We are then asked which user account we want to connect the mailbox. We still have the option here of going with Sarah Gibbs. Or by selecting **Connect to the following user account** and clicking the **Browse** button we could pick another user (Figure 6-10). The only user accounts displayed will be those that do not have a mailbox. When satisfied click **Finish**. The mailbox will appear under the mailboxes tab but as the new user account.

It's worth noting that if you do not see the option **Yes, connect to the user account above** then the original user account no longer exists. In that case your only option available will be **No, I want to connect to a different user account**.

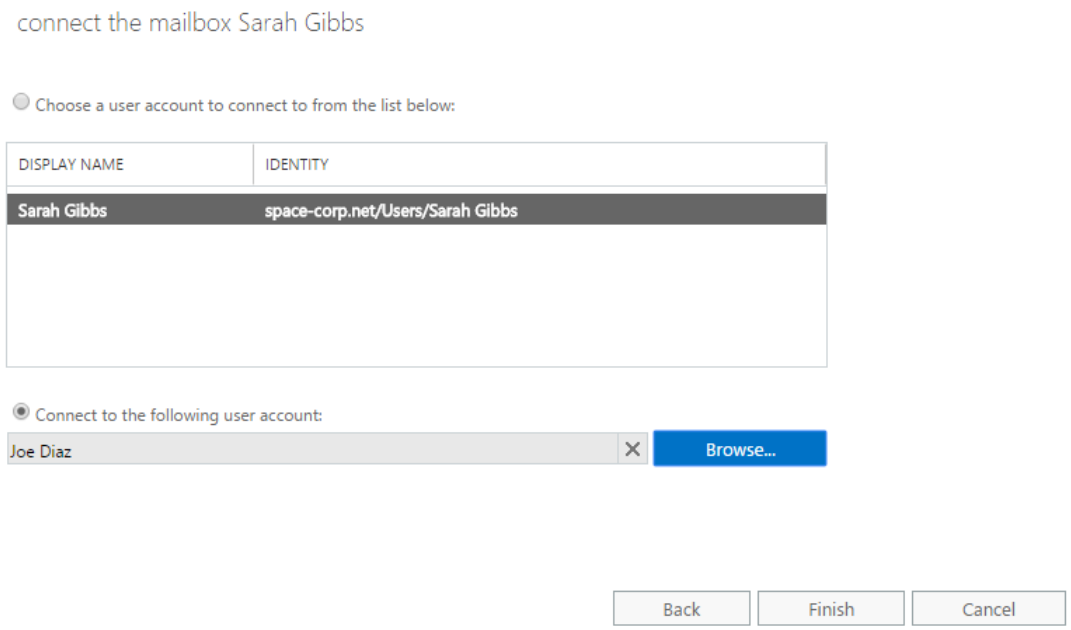

Figure 0-10: Choosing the owner of a disconnected mailbox

In our example, we connected Sarah Gibbs' back to her original mailbox.

Before we examine the process of reconnecting a mailbox with EMS let's have a little fun. Let's look for all the disconnected mailboxes in our environment. A great way to do this is to search for all mailboxes that have a disconnect date timestamp. For this query, we will need to use the **Get-MailboxStatistics** parameter.

```
[PS] C:\> Get-MailboxStatistics -Filter "DisconnectDate -ne `$null" -Server "EXCH02"
```

In this command, we look for all mailboxes that have a disconnect date timestamp. A normal mailbox would not have a value in this field. It would be null. This query specifically returns all mailboxes that have a value other than $null. In our example, our query returns Sarah Gibbs.

```
DisplayName           ItemCount           StorageLimitStatus                LastLogonTime
-----------           ---------           ------------------                -------------
Sarah Gibbs                   9                                   12/31/2016 2:52:52 PM
```

Let's modify our original command a little further. While the default response brings back some great information we can take it one step further and gain an even richer response. Let's select which fields we want to be in our response. We will then need to pipe that into a **Format-Table**.

```
[PS] C:\> Get-MailboxStatistics -Filter "DisconnectDate -ne `$null" -Server "EXCH02" | Select
DisplayName, Database, ItemCount, TotalItemSize, DisconnectDate | Format-Table -AutoSize

DisplayName    Database       ItemCount            TotalItemSize                DisconnectDate
-----------    --------       ---------            -------------                --------------
Sarah Gibbs        DB02               9       45.53 KB (46,626 bytes)       12/31/2016 4:38:44 PM
```

In this command, we specify the fields we want the query to return by using the **Select** statement. We then list those fields in a comma formatted list. We finish our command by piping those selections into the Format-Table command. While the new fields like DisconnectDate, or TotalItemSize are certainly more useful the most important field we gain is Database. This value is required when we reconnect the mailbox to a user.

Real World: If you are ever curious what fields you can specify with the Select statement, first run your command by piping it into Format-List. Every field Format-List returns can be used with the Select statement.

Now let's look at how we reconnect a mailbox back to its original owner. To do this we will need to issue the **Connect-Mailbox** command. In this example, we will reconnect Sarah Gibbs to her mailbox.

```
[PS] C:\> Connect-Mailbox -Database "DB02" -Identity "Sarah Gibbs"

Confirm
Do you want to connect this mailbox to the user account "space-corp.net/Users/Sarah Gibbs" and use
the alias "SarahGibbs"?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [?] Help (default is "Y"):
```

The command identifies that the original user still exists and asks us if we want to reattach the mailbox with the original owner. If we press **Enter** the default value of yes is accepted and the mailbox is reconnected to Sarah's user account. If we type **N** for no and press **Enter** the command cancels out and the mailbox will remain disconnected.

If we want to attach Sarah's mailbox to another user (who is not mailbox-enabled) we would specify the **-User** parameter. In this example, we connect Sarah Gibbs' mailbox to Joe Diaz. Joe does not have a mailbox of his own.

```
[PS] C:\> Connect-Mailbox -Database "DB02" -Identity "Sarah Gibbs" -User "Joe Diaz"
```

Unlike last time there is no prompt to confirm. Instead Sarah's mailbox is reassigned to Joe Diaz immediately. Sarah no longer has access to her mailbox.

Note: For an extensive list of all available parameters for **Connect-Mailbox** cmdlet check the following article https://technet.microsoft.com/en-us/library/aa997878(v=exchg.160).aspx

Real World: In prior versions of Exchange after a mailbox had been disconnected it was necessary to run the Clean-MailboxDatabase command. This cmdlet is no longer necessary. With Exchange 2016 a clean-up process is automatically executed after a mailbox is disconnected. If you do experience problems with the clean-up process you can leverage the Update-StoreMailboxState cmdlet to force an immediate sync. For more info check this article https://technet.microsoft.com/en-us/library/jj860462(v=exchg.160).aspx

Moving data from a disabled mailbox to an active mailbox

Let's consider a scenario where another user needs access to the data from a disabled mailbox. However, that user already has a mailbox of their own. One option could be to reconnect the mailbox as a shared mailbox and grant our user full access. Another option could be to migrate the data from the disabled mailbox to the target user's active mailbox.

In this section, we explore the process of moving mailbox data from a disabled mailbox to an active mailbox. There is no method for doing this in the EAC so let's jump straight to EMS. To perform the data move we need to determine which database hosts the disabled mailbox. This requires the following query.

```
[PS] C:\> Get-MailboxDatabase | Get-MailboxStatistics | Where {$_.DisconnectReason -eq "Disabled"} |
Format-List DisplayName,Database

DisplayName : Brian Best
Database    : DB02
```

This command looks for all mailboxes with a disconnected reason of disabled across all databases. It then pipes those results into a formatted list returning only the fields *DisplayName* and *Database*. In our example, we find that Brian Best's disabled mailbox is stored on DB02. With the database located we can now issue the **New-MailboxRestoreRequest** command to start the data move.

```
[PS] C:\> New-MailboxRestoreRequest -SourceStoreMailbox "Brian Best" -SourceDatabase "DB02" -
TargetMailbox "Lynn Simmons" -TargetRootFolder "Imported" -AllowLegacyDNMismatch

Name                                               TargetMailbox                                 Status
----                                               -------------                                 ------
MailboxRestore                                     space-corp.net/Users/Lynn Simmons             Queued
```

In this cmdlet:

-SourceStoreMailbox specifies the disabled mailbox to migrate data from. You can use several fields to identify the source mailbox including DisplayName, MailboxGuid and LegacyDN. In our example, we went with DisplayName.

-SourceDatabase specifies the original database of the disabled mailbox.

-TargetMailbox specifies where we want to move the data.

-TargetRootFolder is optional. Without this parameter, the contents of the two mailboxes would be more intricately merged. The data from duplicate folders such as the Inbox, Calendar, or, Sent Items would be combined. With the TargetRootFolder parameter we can import the data under a common root folder. This maintains a separate folder tree for the imported data. In our example, we specified a root folder of *Imported* (Figure 6-11). The benefit is that the user can easily determine what was imported.

-AllowLegacyDNMismatch is necessary in this type of restore because the source and target LegacyExchangeDN's will not match. Adding this parameter instructs the New-MailboxRestoreRequest command to disregard any errors caused by the mismatch.

To verify the status of our data move, we can issue the **Get-MailboxRestoreRequest** command.

```
[PS] C:\> Get-MailboxRestoreRequest

Name                           TargetMailbox                           Status
----                           -------------                           ------
MailboxRestore                 space-corp.net/Users/Lynn Simmons       Completed
```

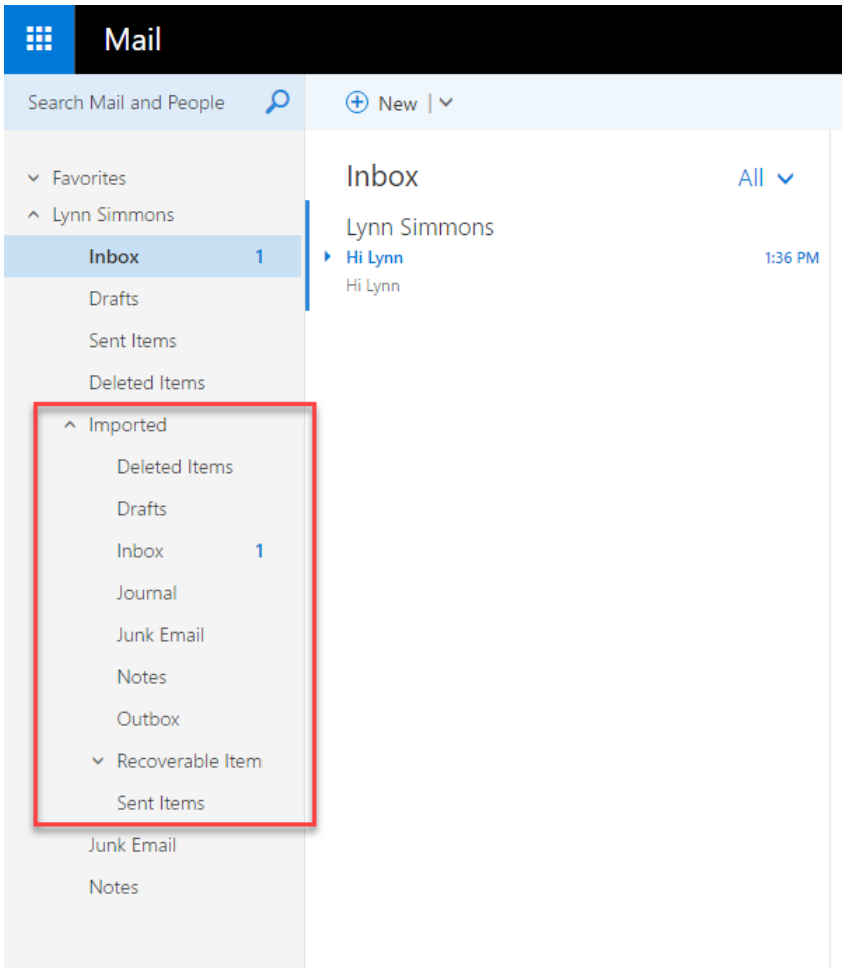

Figure 0-11: Imported mailbox data with -TargetRootFolder parameter

For additional detail, we can pipe the **Get-MailboxRestoreRequest** command into **Get-MailboxRestoreRequestStatistics**. Adding the **–IncludeReport** parameter adds an even greater level of detail. Excellent when troubleshooting why a restore has failed.

```
[PS] C:\> Get-MailboxRestoreRequest -Name "MailboxRestore" | Get-MailboxRestoreRequestStatistics -IncludeReport | Format-List
```

The report can get quite long so I have omitted that output from the book.

Note: For an in-depth look at the **New-MailboxRestoreRequest** cmdlet check the following article https://technet.microsoft.com/en-us/library/ff829875(v=exchg.160).aspx

Permanently deleting a mailbox

As mentioned in prior sections once a mailbox is disconnected it can be reconnected if the mailbox retention period has not expired. Once the retention period expires the mailbox and all its data is permanently deleted from the database. However, you may wish to skip the mailbox retention period entirely and permanently delete the mailbox.

Warning: Permanently deleting a mailbox immediately removes that mailbox and its data from the database. The mailbox cannot be reconnected. Should you need that mailbox you will need to recover it from either a lagged database copy or traditional backup.

The process of immediately and permanently deleting a mailbox is not present in the EAC. For this process, you will have to use PowerShell.

To immediately and permanently delete a mailbox and its user account the following command would be needed.

```
[PS] C:\> Remove-Mailbox -Identity "Lynn Simmons" -Permanent $true

Confirm
Are you sure you want to perform this action?
Removing the mailbox "Lynn Simmons" will remove the Active Directory user object and mark the
mailbox in the database for removal.
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [?] Help (default is "Y"):
```

The **–Permanent** parameter is not available on the Disable-Mailbox cmdlet. To permanently delete a mailbox but maintain the user you will need to perform three steps. The first is to disconnect the mailbox from the user. We have covered this in depth earlier in this chapter but here is that command again.

```
[PS] C:\> Disable-Mailbox -Identity "Lynn Simmons"

Confirm
Are you sure you want to perform this action?
Disabling mailbox "Lynn Simmons" will remove the Exchange properties from the Active Directory user
object and mark the mailbox in the database for removal. If the mailbox has an archive or remote
archive, the archive will also be marked for removal. In the case of remote archives, this action is
permanent. You can't reconnect this user to the remote archive again.

[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [?] Help (default is "Y"):
```

The second is we need to find out what database hosted Lynn's mailbox and the reason why her mailbox was disconnected. Both are required parameters for the third command. To find the database and disconnect reason issue the following query.

```
[PS] C:\> Get-MailboxDatabase | Get-MailboxStatistics | Where {$_.DisplayName -eq "Lynn Simmons"} |
Format-List DisplayName,Database,DisconnectReason


DisplayName          : Lynn Simmons
Database             : DB02
DisconnectReason     : Disabled
```

Disabled versus Soft deleted: There are two possible disconnect states for a mailbox: *Disabled and Soft-Deleted*. When you disable or delete the mailbox it is put into a d*isabled* state. When a mailbox is moved to a different database the original source data is put into a s*oft-deleted* state. This source data can be leveraged if the mailbox move experienced corruption and that data needs to be retrieved.

The third step is to permanently delete the disconnected mailbox from the store. To do this we would issue the **Remove-StoreMailbox** command.

```
[PS] C:\> Remove-StoreMailbox -Database DB02 -Identity "Lynn Simmons" -MailboxState Disabled

Confirm
Are you sure you want to perform this action?
Removing mailbox "Lynn Simmons" on database "DB02".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [?] Help (default is "Y"):
```

In this command:

-Database specifies the database that hosts the disconnected mailbox

-Identity specified the identity of the mailbox to be removed

-MailboxState specifies whether the mailbox is in a *disabled* or *soft-deleted* state. Specifying the incorrect state will cause the command to fail.

Moving a mailbox to a different database

It is not uncommon to have multiple databases in your Exchange environment. The need for multiple databases may arise as part of a high availability design, or, maybe the result of a recovery time objective (RTO) that dictates the maximize size for a database.

The method that users are distributed across multiple databases could be the result of any number of factors. Users may be assigned to databases purely based on the Exchange balancing algorithm, allowing Exchange free reign to distribute mailboxes as they are created. On the other hand, mailbox location could be tightly dictated by the administrator. For example, an administrator may distribute mailboxes per job function, last name, or, geographic region. This could create a scenario where an administrator will need to move a mailbox when any of these conditions change such as a name or location.

An administrator may also need to move a mailbox to a new database if a database is deemed too large. Or it may happen to shrink the data footprint if it is believed a lot of white space exists in the current database.

When considering defragmenting a database it is considerably easier--and preferred--to move those users to a new database and delete the old database. Not only does this eliminate massive user downtime (because the database is never taken offline) it also won't require 110% of free disk space that a defragment requires. It is also worth mentioning that moving users does not eliminate the white space in a database.

Moving mailboxes to a new database also strips away corruption. Corrupted items are not moved. This can be a great troubleshooting step when rectifying bad items in mailboxes.

Whatever the reason it is more than likely an administrator will be tasked with moving mailboxes between databases. In this section, we explore how to do this with both the EAC and EMS. Let's get started.

Real World: In the EAC you can add and remove columns from view. One column I like to add is the databases column. To add the database column, click the **More** button (represented by an ellipsis) and select **Add/Remove Columns**. From here select **Database** and click **Ok**. This provides a column that indicates which database each user belongs to.

In the EAC select the **Recipients** tab on the left and **Mailboxes** sub tab across the top. Select the mailbox you wish to move. In the action pane scroll to the bottom and click the **To another database** link.

On the *New Local Mailbox Move* window (Figure 6-12) type a descriptive batch name in the **New migration batch name** field. Then specify if you want to move the user's primary, archive, or, both mailboxes. Under *Target database* click the **Browse** button and select where you want to move the primary mailbox. Similarly, under *Target archive database* click the **Browse** button and select where you want to move the archive mailbox. The browse buttons will deactivate based on your selection under *Archive*.

The **Bad Item Limit** field is the amount of loss you are willing to accept. The default value of 10 indicates that the move will continue if it encounters up to 10 corrupt items. If the migration encounters 11 corrupt items, the batch will fail. Setting a value of 0 instructs Exchange to accept no data loss. Click **Next**.

On the *Start the batch* page you can specify which user should be emailed the migration report. To select a user, click the **Browse** button.

By default, the migration is set to start as soon as you complete the wizard. If you want to execute the batch later select **Manually start the batch later**, otherwise select **Automatically start the batch**.

The migration batch is also set to **Automatically complete the migration batch** which copies all user data to the target database, updates the user in Active Directory and, soft deletes the mailbox in the source database. Completing the batch can result in a tiny blip of downtime which may be undesirable to your users. In this case, you may wish to select **Manually complete the batch**. The difference between the two is that manually complete the batch copies all mailbox data to the target database but doesn't update the user's attributes in Active Directory. Nor does it soft delete the mailbox. The user is still accessing their data from the source

database. The batch stops at 95% which is the end of the copy process. The batch will remain at 95% until it is manually completed. Incremental syncs between the source and target databases will occur every 24 hours. One final sync is performed when the migration batch is manually completed.

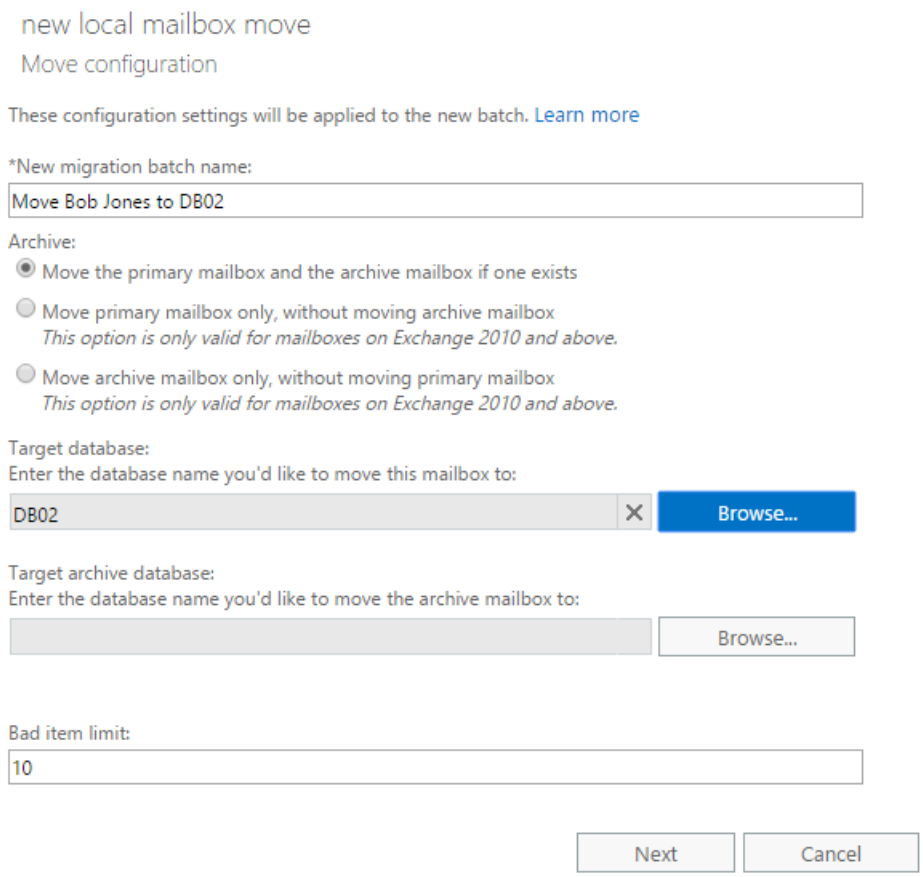

Figure 0-12: New Migration Batch

Click **New** to save the batch. An information dialog will ask you if you want to go to the migration dashboard to view the status of you batch. Select **Yes** to be automatically switched to the **Migration** tab. If you pick **No,** you can find your migration batch later by navigating to the **Recipients** tab on the left and **Migration** sub tab across the top (Figure 6-13).

To manually start a batch, select the migration batch with a status of *Created* and click **Start** (represented by a triangle button). Click **Yes** to confirm.

To manually complete a batch, select the migration batch with a status of *Synced* and click **Complete this migration batch** in the action pane. Click **Yes** to confirm.

To stop a batch, select the **Stop** button (represented by a square). Click **Yes** to confirm.

Some basic statistics are shown in the action pane. To get a more detailed look into the migration and to download logs click the **View details** link.

Once a batch is completed (and the migration statistics and logs are no longer needed) it can be safely deleted. To delete a batch, select the **Delete** button (represented by a trash can). Click **Yes** to confirm. Note that only migrations that have a *Completed* or *Stopped* state can be deleted.

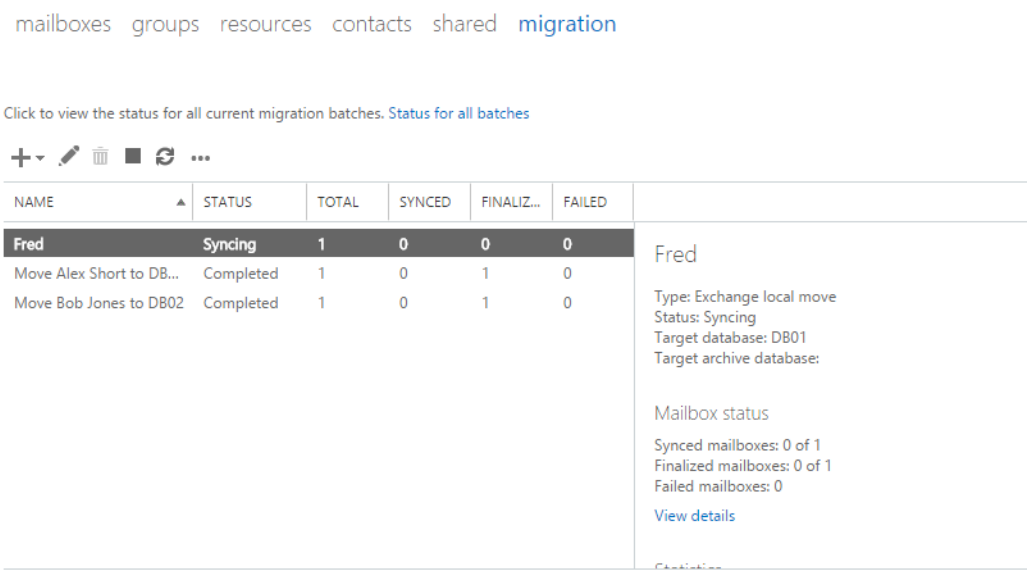

Figure 0-13: Managing the Migration Batch

Let's explore this same process but in EMS. For this we will use the **New-MigrationBatch** command.

> **Note:** New-MigrationBatch can be used for several different purposes, including on-boarding and off-boarding mailboxes to Exchange Online. Be sure to check *Chapter 13: Updates and Migration* for more on on-boarding and off-boarding to Office 365.

First, we will need to capture the email addresses of the users we want to move in a CSV file. The column heading must read "Email Address" (Figure 6-14). In this example, we are going to move user Kim Steele.

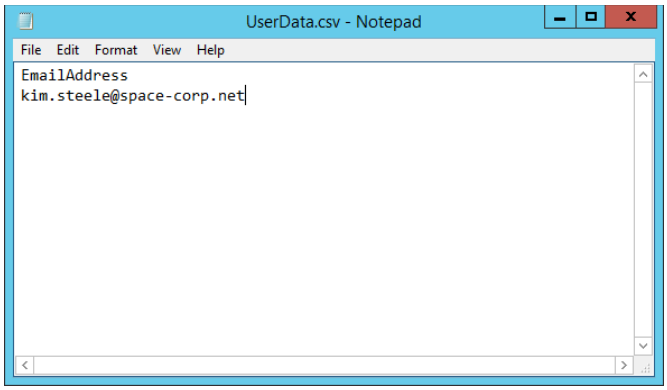

Figure 0-14: CSV File Requirements for New-MigrationBatch

We then parse that CSV file into our New-MigrationBatch command.

```
[PS] C:\> New-MigrationBatch -Name "Move Kim Steele to DB02" -CSVData
([System.IO.File]::ReadAllBytes("C:\MoveRequest\UserData.csv")) -Local -TargetDatabases DB02 -
AutoStart -AutoComplete -BadItemLimit 10 -NotificationEmails administrator@space-corp.net

Identity                    Status      Type                TotalCount
--------                    ------      ----                ----------
Move Kim Steele to DB02     Syncing     ExchangeLocalMove   1
```

In this command:

-Name gives the migration batch a friendly name. Its best to go with something descriptive here.

-CSVData specifies the path to the CSV file.

-Local identifies the migration batch as a local move request between on-prem databases.

-TargetDatabases specifies where we want to move our users.

–BadItemLimit specifies how much data loss we are willing to accept before the batch fails. The value of 10 identifies we are willing to accept 10 corrupt items to consider the move a success.

–NotificationEmails identifies who should receive a report regarding the success or failure of the batch.

The **–AutoStart** parameter starts the migration batch starts as soon as it is created. Omitting this parameter allows us to start the migration batch later. To start a batch that was missing -AutoStart, we would issue the following command.

```
[PS] C:\> Start-MigrationBatch -Identity "Move Kim Steele to DB02"
```

Similarly, the **–AutoComplete** parameter finalizes the migration batch as soon as the copy process is complete. Omitting this parameter copies all mailbox data to the target database but doesn't update the user's attributes in Active Directory. The batch will remain at 95% until it is manually completed. To finalize a batch that was missing -AutoComplete, we would issue the following command.

```
[PS] C:\> Complete-MigrationBatch -Identity "Move Kim Steele to DB02"
```

If we had multiple migration batches to start, we can pipe the results from **Get-MigrationBatch**.

```
[PS] C:\> Get-Migrationbatch | Start-MigrationBatch
```

Or if we had multiple batches to complete.

```
[PS] C:\> Get-Migrationbatch | Complete-MigrationBatch
```

To get progress details on the move we would issue the **Get-MigrationUser** command.

```
[PS] C:\> Get-MigrationUser -Batch "Move Kim Steele to DB02"

Identity                    Batch                       Status        LastSyncTime
--------                    -----                       ------        ------------
kim.steele@space-corp.net   Move Kim Steele to DB02     Completed     1/15/2017 11:52:29 AM
```

For additional detail, we can pipe the **Get-MigrationUser** command into **Get-MigrationUserStatistics**. Adding the **–IncludeReport** parameter adds an even greater level of detail. Excellent when troubleshooting why a move has failed.

```
[PS] C:\> Get-MigrationUser -Batch "Move Kim Steele to DB02" | Get-MigrationUserStatistics -
IncludeReport | Format-List
```

The report can get quite long so I have omitted that output from the book.

Once a migration batch has completed and we no longer need the statistics or move report we can issue the **Remove-MigrationBatch** command.

```
[PS] C:\> Remove-MigrationBatch -Identity "Move Kim Steele to DB02"

Confirm
Are you sure you want to perform this action?
Remove the migration batch "Move Kim Steele to DB02"?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [?] Help (default is "Y"):
```

Migration Batch versus Move Request: Exchange 2013 introduced the New-MigrationBatch cmdlet. This command is more feature rich than its older counterpart New-MoveRequest. That said, New-MoveRequest is still a perfectly acceptable method for performing a local move request in Exchange 2016. It is worth noting though that the older method does not allow you to stage data at the target database. Nor, does it allow for incremental syncs or provide notification emails on migration success or failure. For more information on the New-MoveRequest cmdlet check the following article https://technet.microsoft.com/en-us/library/dd351123(v=exchg.160).aspx

Moving multiple mailboxes to a different database

In this section, we look at moving multiple mailboxes using the Exchange Admin Center (EAC). The process for the Exchange Management Shell is identical to that of the prior section.

In the EAC select the **Recipients** tab on the left and **Migration** sub tab across the top. Click the **New** button (represented by a plus sign) and select **Move to a different database**.

From the *New Local Mailbox Move* dialog you can select the users directly from Active Directory by choosing **Select the users that you want to move** and click the **Add** button, or, you can import those users from a file by selecting **Specify the users with a CSV file** and clicking **Choose File** (Figure 6-15). In our example, we will pick the former. With the method selected click **Next**.

On the *Move Configuration* page (Figure 6-15) type a descriptive batch name in the **New migration batch name** field. Then specify if you want to move the user's primary, archive, or, both mailboxes. Under *Target database* click the **Browse** button and select where you want to move the primary mailbox. Similarly, under *Target archive database* click the **Browse** button and select where you want to move the archive mailbox.

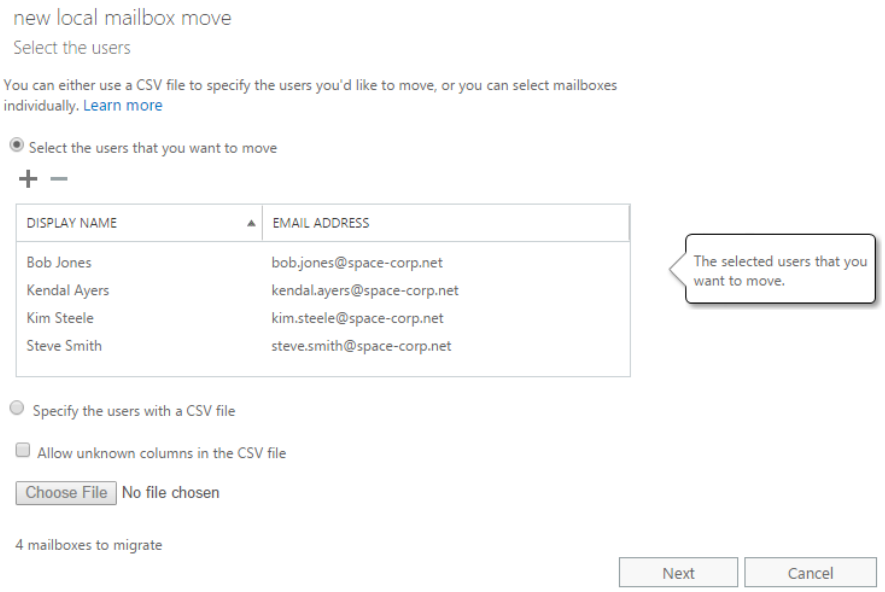

Figure 0-15: Moving multiple mailboxes to a different database

The **Bad Item Limit** field is the amount of data loss you are willing to accept. The default value is 10 items per mailbox. Each mailbox will continue to copy if it encounters up to 10 bad items. If one of the mailboxes encounters 11 bad items only that mailbox will fail. All other mailboxes in the batch will continue to copy if their bad item count remains below the threshold you set. Click **Next**.

On the *Start the batch* page you can specify which user should be emailed the migration report. To select a user, click the **Browse** button.

Pick whether you want to **Automatically start the batch** once the wizard completes otherwise select **Manually start the batch later**. Pick whether you want to **Automatically complete the migration batch** once all data has been copied to the target database otherwise select **Manually complete the batch**. For details on these options refer to the previous section.

Click **New** to save the batch. You will be return to the migration tab. Depending on your choices you may need to start the batch by selecting the batch and clicking the **Start** button.

Creating an archive mailbox

As an administrator, you can assign a user an archive mailbox. The archive can either sit in the same database as the user's primary mailbox, or, in a completely different database. This is called an In-Place Archive.

One of the early business cases for the archive was to move old and infrequently accessed mail to lower tier storage, in turn eliminating the need for PST files. Whereas newer more frequently accessed mail would stay on faster top tier storage. With each release of Exchange reducing the necessary disk IOPS the need for top tier storage is a thing of the past. This could bring into question why an archive is needed at all. Especially as the cost per gigabyte (or terabyte) of entry-level and midline drives plummets allowing for much higher mailbox quotas. In this section, we will look at creating an archive with both the EAC and EMS.

Select the **Recipients** tab on the left and **Mailboxes** sub tab across the top. Select the mailbox you wish to create an archive for. In the action pane under *In-Place Archive* select **Enable** link.

From the *Create In-Place Archive* dialog select the database for the archive by clicking the **Browse** button. Click **Ok**.

Back on the mailboxes tab you will now see the word archive in parenthesis under the *Mailbox Type* column for that user. This identifies that the user has an archive. In the action pane, you will have the option to disconnect the archive by clicking the **Disable** link. The archive mailbox follows the same mailbox retention period as the primary mailbox. The default is 30 days. To reconnect an archive, navigate to **More > Connect a mailbox**. A word of warning; using the enable link a second time will create a new empty archive for the user.

Selecting **View Details** brings up a dialog that allows you to change the display name of the archive (this is what the user will see), the quota for the archive and current archive usage.

Let's explore how to enable an archive in EMS. Using Steve Smith as an example our command would look like this.

```
[PS] C:\> Enable-Mailbox -Identity "Steve Smith" -Archive -ArchiveDatabase DB01

Name                  Alias                 ServerName            ProhibitSendQuota
----                  -----                 ----------            -----------------
Steve Smith           steve.smith           exch01                Unlimited
```

Where:

-Identity specifies the user we want to enable an archive for.

-Archive specifies what we are enabling is an archive

-ArchiveDatabase specifies where the archive mailbox should be created.

If we wanted to create an archive for any user that currently does not have an archive, we could use a filter. In this command, we filter for any user mailboxes with an archive state of none.

```
[PS] C:\> Get-Mailbox -Filter {ArchiveState -eq "None" -and RecipientTypeDetails -eq "UserMailbox"}
| Enable-Mailbox -Archive -ArchiveDatabase DB02

Name                      Alias                 ServerName        ProhibitSendQuota
----                      -----                 ----------        -----------------
Administrator             Administrator         exch01            Unlimited
Sarah Gibbs               SarahGibbs            exch02            Unlimited
Lynn Simmons              LynnSimmons           exch01            Unlimited
```

Most of the commands that work for the primary mailbox should also work for the archive mailbox. It is often a matter of adding the **–Archive** parameter to signify you are working with the archive and not the primary mailbox. In the example below we modify the Get-Mailbox command with the –Archive switch to only return information about archive mailboxes in our organization. If in doubt check the TechNet documentation on proper command usage.

```
[PS] C:\> Get-Mailbox -Archive

Name                  Alias           ServerName      ProhibitSendQuota
----                  -----           ----------      -----------------
Steve Smith           steve.smith     exch01          Unlimited
Kim Steele            kim.steele      exch01          Unlimited
```

To disable an archive for a user we would issue the following command. You will notice the only difference between this and disabling the primary mailbox is the addition of the –Archive parameter.

```
[PS] C:\> Disable-Mailbox -Identity "Rachel Hughes" -Archive

Confirm
Are you sure you want to perform this action?
Disabling the archive for "Rachel Hughes" will remove the archive for this user and mark it in the
database for removal.
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [?] Help (default is "Y"):
```

Creating an online archive for an on-premises mailbox

If you have a hybrid connection with Office 365 you get the additional option of moving your archive off-prem. You will need a license for Exchange Online Archive for this task. To configure an Exchange Online Archive for an on-prem mailbox complete the following tasks.

From the **Recipients** tab select the mailbox you wish to enable for online archive. From the action pane click **Enable** under *In-Place Archive*. From the *Create In-Place Archive* dialog select **Cloud-based archive: <tenant>**. Click **Ok**.

If you select **View Details** the *Status* field will report *Cloud-based archive pending* until directory synchronization executes and Exchange Online provisions the archive. This may take a couple of hours. Once provisioned the status will change to *Cloud-based archive created*. You will also notice that the archive quotas cannot be changed (Figure 6-16).

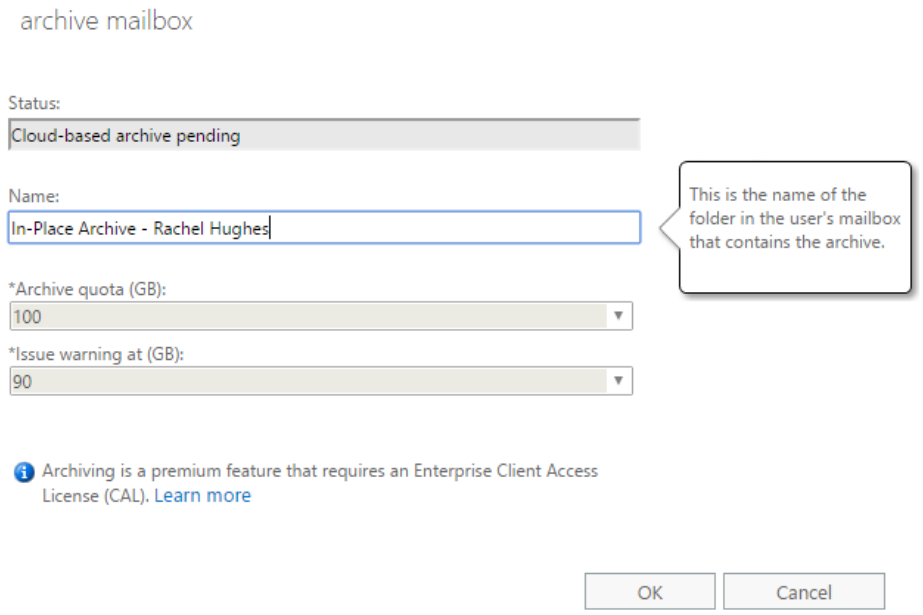

Figure 0-16: Managing an Exchange Online Archive

To configure an Exchange Online Archive with EMS you would issue the following command.

```
[PS] C:\> Enable-Mailbox -Identity "Rachel Hughes" -RemoteArchive -ArchiveDomain "space-
corp.mail.onmicrosoft.com"
```

In this command the **–RemoteArchive** parameter signifies this archive will be in Office 365. This is a direct replacement of the –Archive parameter which creates a local archive. The **–ArchiveDomain** parameter specifies your Office 365 service domain, which is typically in the form of *<tenant>.mail.onmicrosoft.com*.

Managing mailbox properties

So far in this chapter we have discussed how to manipulate mailboxes from creation to deletion. Let's now look at the managing some of the properties of a mailbox. We will look at those first through the EAC.

To access the properties of a mailbox, select the **Recipients** tab on the left and **Mailboxes** sub tab across the top. From here you can either double click the mailbox you want to edit, or, select the mailbox and click the **Edit** button (represented by a pencil).

This will launch a dialog with several tabs. The first tab is the **General** tab. A lot of the information here can also be found in Active Directory Users and Computers (ADUC) such as first name, last name, user logon name and the option to change password at next logon. One option here that is not present in ADUC is **Hide from address lists**. Checking this option does exactly what the name implies. It will hide that user from all address lists in the address book, including the global address list (GAL). Users will not be able to search for this user. Clicking the **More options** link displays a couple of non-editable fields such as which organization unit (OU) and database the user belongs. It also displays the *Custom Attributes* box. From here you can select the **Edit** button and add up to 15 custom attributes. These attributes can be useful when developing filters for Email Address Policies or custom address lists.

The next tab is **Mailbox Usage**. This tab lists the users last logon time and how much of their current mailbox quota they have consumed. Note that this only displays the quota for the primary mailbox. Refer to the last section on viewing archive consumption. Selecting **More options** allows us to modify the quota and retention periods for this mailbox. By default, the quotas and retention periods are set at the database level. Making changes at the mailbox level supersedes those settings.

To set a quota at the mailbox level select **Customize the quotas settings for this mailbox** and modify the **issue a warning**, **prohibit send** and **prohibit send and receive** fields to the desired value in gigabytes.

To change the default retention period at the mailbox level, select, **Customize the retention settings for this mailbox** and enter a value in the **Keep deleted items for day(s)** field. Once a deleted item surpasses this threshold it is moved from the recovery folder to the purges folder (hard deleted). The mailbox assistant removes any items in the purges folder from the database. The exception is if the mailbox is under In-Place hold. If the mailbox is under an In-Place hold that item is moved from the recovery folder and maintained in the purges folder for the duration of the hold. You can also select to keep deleted items until a successful backup of the Exchange databases has been completed.

Real World: Before changing a mailbox quota or retention period it is a good idea to revisit your Exchange server design to determine if any capacity needs to be added.

All the fields under the **Contact Information** and **Organization** tabs can be found in ADUC. Any information entered here will be displayed in the user's account properties in ADUC and vice versa.

The **Email Address** tab lists all addresses assigned to the mailbox. By default, addresses populated here are assigned by an email address policy (Figure 6-17). Deselecting the checkbox **Automatically update email addresses based on the email address policy** will block this mailbox from retrieving its addresses from the email address policy. Keep in mind that any addresses already assigned by a policy will not be removed when this checkbox is unchecked.

The user's primary address is highlighted in bold. This is the address other users see and is also referred to as the reply address. The email address policy governs the reply address. To change the reply address you will first need to deselect **Automatically update email addresses based on the email address policy**. Then select the address you want to make the reply address and click the **Edit** button. Check the box **Make this the reply address**. Click **Ok**. This address now becomes the user's primary address and is highlighted in bold. Note that the checkbox to change the reply address is only visible when editing secondary email addresses.

To add an email address, click the **Add** button. On the *New Email Address* dialog specify the *email address type* (in most cases this will be SMTP). Then specify the address in **Email Address** field. Click **Ok**.

To remove an email address, select the address from the list and click **Remove.** Note that only secondary addresses can be removed. If you want to remove an address that is currently the primary address you will first need to assign another primary address.

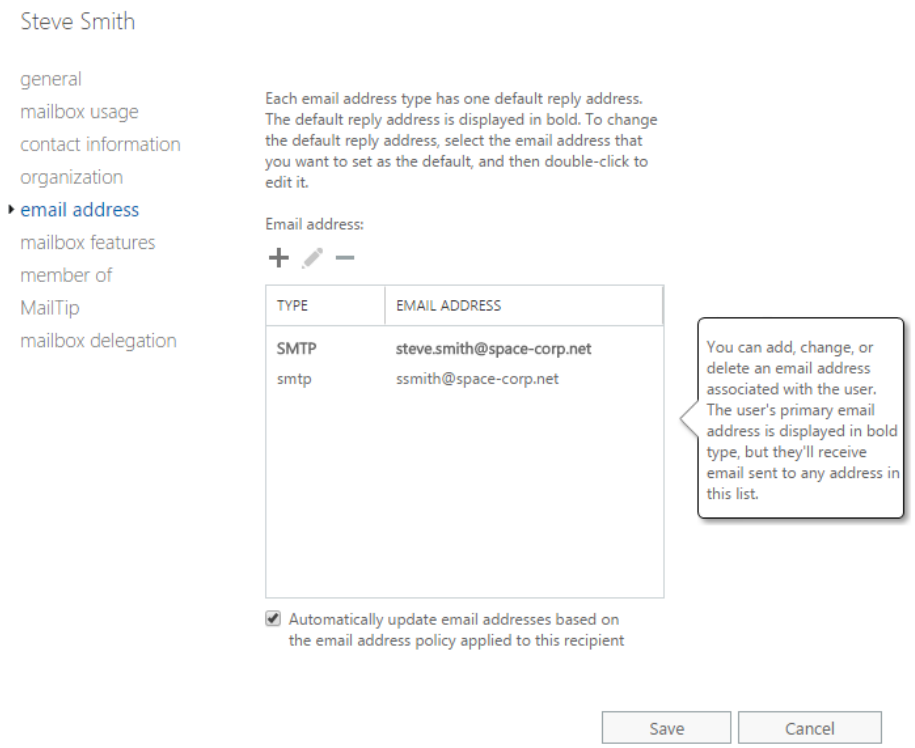

Figure 0-17: Modifying email addresses assigned to a mailbox

The **Member Of** tab is a read-only list of all mail-enabled groups the user belongs to. This includes all distribution groups and mail-enabled security groups. Dynamic distribution groups are not shown here.

The **MailTip** tab allows you to set custom notifications that alert senders when they add this mailbox as a recipient to an email message. This mail tip is shown above the recipient line. An example MailTip could be "Please allow up to 2 business days for a response."

When configuring these properties through EMS there are a few common commands. Those are Set-User, Set-Mailbox and, Set-CASMailbox.

Set-User is an Active Directory command and manages attributes for the user account. For example, if we need to change a user's first name, middle initial or last name, we would use **Set-User**. In this example, we change Steve's first name to Steven and add a middle initial. We also update his display name in Active Directory through the -Name parameter.

```
[PS] C:\> Set-User -Identity "Steve Smith" -FirstName "Steven" -Initials "J" -LastName "Smith" -Name
"Steven Smith"
```

To change Steve's location and contact information, we would issue a command such as this.

```
[PS] C:\> Set-User -Identity "Steve Smith" -StreetAddress "400 Rocket Avenue Suite 101" -City
"Merritt Island" -StateOrProvince "FL" -PostalCode "32953" -CountryOrRegion "US" -Phone "555-555-
0122" -MobilePhone "555-555-0123" -Fax "555-555-0124" -Office "555-555-0125" -HomePhone "555-555-
0126" -WebPage http://space-corp.net
```

To changes Steve's organization information, we would issue a command like this.

```
[PS] C:\> Set-User -Identity "Steve Smith" -Title "Director of IT" -Department "Information
Technology" -Company "Space Corp" -Manager "Harvey Lovell"
```

On the other hand, if we want to change mail attributes we use the **Set-Mailbox** command. For example, to change Steve's alias, also known as the mail nickname, we would issue this command.

```
[PS] C:\> Set-Mailbox -Identity "Steve Smith" -Alias "Steven.J.Smith"
```

To configure a mailbox quota for Steve our **Set-Mailbox** command would look like this.

```
[PS] C:\> Set-Mailbox -Identity "Steve Smith" -IssueWarningQuota 2GB -ProhibitSendQuota 3GB -
ProhibitSendReceiveQuota 4GB -UseDatabaseQuotaDefaults $false
```

For the quota values, we can specify bytes (B), kilobytes (KB), megabytes (MB), gigabytes (GB), or, terabytes (TB). If we neglect to add the unit of measure the value will be treated as bytes. It is also required to configure the parameter **-UseDatabaseQuotasDefault** to *$false*. Otherwise the values we enter will be ignored. In fact, Exchange will issue a warning if this parameter is not present. If you ever need to review a user's quota you can issue the Get-Mailbox command. In this example, we instruct EMS to only return the quota fields we are interested in.

```
[PS] C:\> Get-Mailbox -Identity "Steve Smith" | Format-List Name, IssueWarningQuota,
ProhibitSendQuota, ProhibitSendReceiveQuota, UseDatabaseQuotaDefaults

Name                      : Steven Smith
IssueWarningQuota         : 2 GB (2,147,483,648 bytes)
ProhibitSendQuota         : 3 GB (3,221,225,472 bytes)
ProhibitSendReceiveQuota : 4 GB (4,294,967,296 bytes)
UseDatabaseQuotaDefaults : False
```

To configure email addresses for Steve we would use **Set-Mailbox**. However, we may first want to determine what addresses Steve currently has assigned. To do this we will use the Get-Mailbox command.

```
[PS] C:\> Get-Mailbox -Identity "Steve Smith" | Format-List Name, EmailAddresses

Name           : Steven Smith
EmailAddresses : {SMTP:steven.j.smith@space-corp.net, smtp:steve.smith@space-corp.net}
```

From this output, we can see that Steve has two email addresses of type SMTP assigned. The address with SMTP capitalized identifies which is the primary or reply-to address. All secondary addresses are designated with SMTP in lowercase. To add a secondary email address, we would use the following command. Note how we keep SMTP as lowercase.

```
[PS] C:\> Set-Mailbox -Identity "Steve Smith" -EmailAddresses @{Add="smtp:it.director@space-
corp.net"}
```

To remove a secondary email address, we simply switch add to remove.

```
[PS] C:\> Set-Mailbox -Identity "Steve Smith" -EmailAddresses @{Remove="smtp:it.director@space-
corp.net"}
```

The Email Address Policy (EAP) determines the primary email address. To change the user's primary address, we will need to disable the policy on this mailbox. We do this by setting the **-EmailAddressPolicyEnabled** parameter to *$false*.

```
[PS] C:\> Set-Mailbox -Identity "Steve Smith" -EmailAddressPolicyEnabled $false
```

Next let's change the primary address to steve.smith@space-corp.net. We cannot use the @Add statement because a primary SMTP address already exists and two primary addresses are not allowed. Nor, can we use the @Remove statement because a mailbox must always have a primary address. The easiest way to accomplish this goal is to overwrite the entire email address field. From our previous output, we know that Steve has two email addresses. With that knowledge, our command will look like this. Note that we have capitalized SMTP before our intended primary address.

```
[PS] C:\> Set-Mailbox -Identity "Steve Smith" -EmailAddresses SMTP:steve.smith@space-
corp.net,smtp:steven.j.smith@space-corp.net
```

Note: For an in-depth look at the cmdlets in this section refer to these articles:
Set-User - https://technet.microsoft.com/en-us/library/aa998221(v=exchg.160).aspx
Set-Mailbox - https://technet.microsoft.com/en-us/library/bb123981(v=exchg.160).aspx

Mailbox features

The **Mailbox Features** tab (Figure 6-18) includes many configurable items for a mailbox. In many cases, all the defaults are fine. But you may have business use cases that require some of these options to be changed. In this section, we will explore what each one does.

A **Sharing Policy** dictates what calendar and contact information your users can share with external entities. These could be federated or non-federated entities. Out of the box Exchange ships with a *Default Sharing Policy* that is assigned to all users. This policy can be changed by selecting an alternate policy in the drop-down. This policy and others can be managed through the *Organization* > *Sharing* tabs.

A **Role Assignment Policy** allows a user to self-manage many of their own mailbox properties such as the contact information listed in their Active Directory account. These policies can also allow users to perform other tasks such as the creation and management of their own distribution groups, or, the installation of marketplace apps. Exchange ships with a *Default Role Assignment Policy* that is assigned to all users. The policy can be changed by selecting an alternate policy in the drop-down. This policy and others can be managed through the *Permissions* > *Users Roles* tab. *For more information on role assignment policies refer to Chapter 14.*

A **Retention Policy** is made up of retention tags. These tags dictate how long an item should be maintained in a mailbox and what action should be taken once that item expires. Actions could include archival or deletion. The policy can be changed by selecting an alternate policy in the drop-down. Retention policies can be managed through *Compliance Management* > *Retention Policies*. Retention tags can be managed through *Compliance Management* > *Retention Tags*. *For more information on retention policies refer to Chapter 15.*

An **Address Book Policy** is a way to give a user a custom address book. An Address Book Policy can be incredibly useful when you need to segment the user population. Examples of where this may be necessary could be the result of an acquisition, divestiture, or, a multi-tenant Exchange environment. Exchange does not ship with an Address Book Policy so the drop down will list '*No Policy*'. Address Book Policies can only be created and managed through the Exchange Management Shell. *For more information on Address Book Policies refer to Chapter 9.*

Unified Messaging (UM) allows the user to be enabled for voice messaging features. These features can allow for integration with Skype for Business and other third party products. Selecting **Enable** will launch the *Enable UM Mailbox* wizard. This wizard walks through the process of assigning the user a UM policy, a SIP address (or E.164 number), an extension and PIN. Once the wizard is complete you can revisit those settings by clicking the **View Details** link. Exchange does not ship with any UM policies configured. To configure UM policies and features navigate to the *Unified Messaging* tab. *For more information on Unified Messaging refer to Chapter 17.*

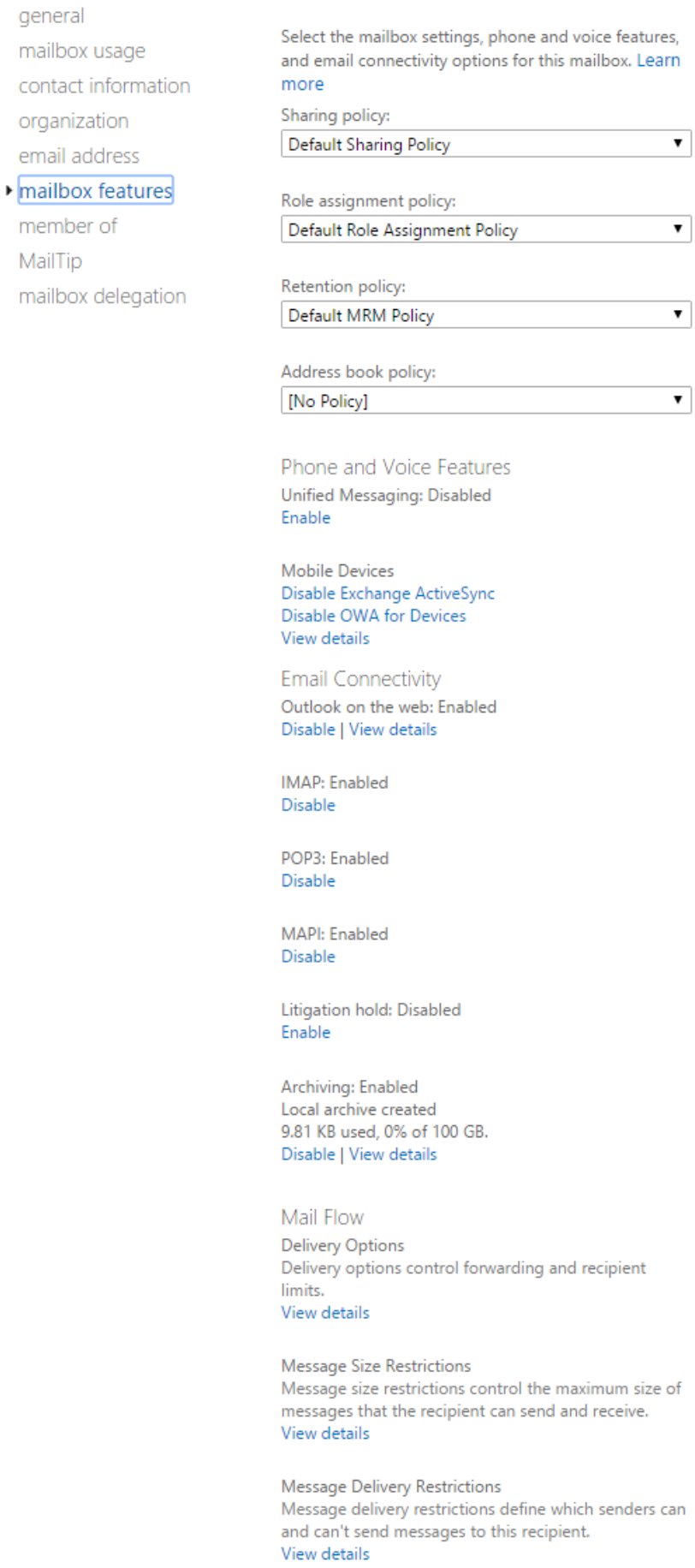

Figure 0-18: Mailbox Feature tab

Mobile devices allow you to control whether the user can make mobile connections to their mailbox. The methods available are ActiveSync and OWA for Devices. Both methods are enabled by default. You can block either method by clicking the disable link.

Selecting **View Details** allows you to select which mobile device mailbox policy to assign to the user. Mobile device policies govern various settings such as device password requirements and wipe on failure settings. Exchange ships with a *Default* policy that can be managed through the *Mobile > Mobile Device Mailbox Policies* tab. The view details tab also allows you to manage your devices, including features such as block device or remote wipe. *For more information on mobile clients refer to Chapter 10.*

Email connectivity allows you to control various access methods to the user's mailbox. These methods include Outlook on the Web (web browser), POP, IMAP and MAPI (Outlook). These methods are all enabled by default. Clicking **View Details** under Outlook on the Web allows an Outlook Web App policy to be assigned to the mailbox. Outlook Web App policies dictate what features a user can access when accessing their mailbox through a web browser. Exchange ships with a default policy. This policy and others can be managed through the *Permissions > Outlook Web App Policies* tabs. *For more information on client connectivity refer to Chapter 10.*

Litigation Hold is disabled by default. Litigation hold retains deleted items. In addition, previous versions of items are also retained. Clicking **Enable** brings up the *Litigation Hold* dialog. From here you can specify the duration of the hold. You can also add a note and URL that will be displayed to the user regarding the reason for the hold. Clicking **Save** enables the hold. You can review and change these settings by clicking the **View Details** link. To disable the hold, click **Disable**. *For more information on litigation hold refer to Chapter 15.*

We have covered archiving earlier in this chapter. Clicking the **Enable** button is another route to create an in-place archive. Once enabled **View Details** displays the current archive usage and allows you to set an archive quota.

Under *Mail Flow* we have **Delivery Options, Message Size Restrictions** and **Message Delivery Restrictions**. Let's review delivery options first. Selecting the **View Details** link under *Delivery Options* allows us to configure message forwarding options and the maximum recipient limit (Figure 6-19).

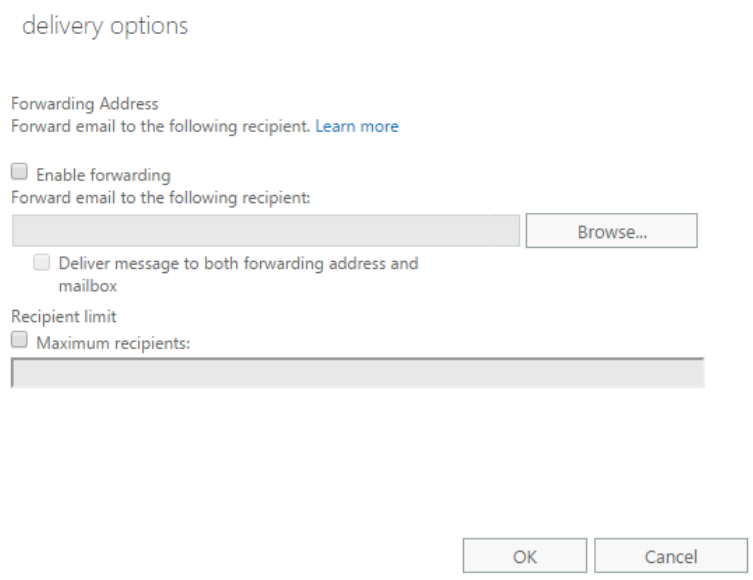

Figure 0-19: Configure mail forwarding and recipient limit

To configure mail forwarding select the **Enable Forwarding** checkbox and click **Browse**. From here select another mail object to forward messages to. If you need to forward to an external email address, that address must be set up as a mail contact in Exchange. We cover the creation of mail contacts in chapter 7. If you want

a copy of the forwarded message to be retained in the original mailbox select **Deliver message to both forwarding address and mailbox**.

If you want to define the maximum number of recipients a user can apply to one message check the box **Maximum Recipients** and specify a number in the field. Click **Ok**.

Selecting **View Details** under *Message Size Restrictions* allows us to control the maximum message size this mailbox can send and receive.

To set the maximum send size click **Maximum message size (KB)** under *Sent Messages* and configure a value in kilobytes. Likewise, to set a maximum receive size click **Maximum message size (KB)** under *Received Messages* and configure a value in kilobytes. Click **Ok**.

Selecting **View Details** under *Message Delivery Restrictions* allows us to dictate who can send messages to this mailbox.

On the *Message Delivery Restrictions* dialog (Figure 6-20), we have the option of configuring an accept list or a reject list. To configure a list of people who can send to this mailbox select **Only senders in the following list** under *Accept Message From*. Click the **Add** button and pick the users who will be allowed to send messages to this mailbox. Note that anyone not in this list will have their message rejected.

Require that all senders are authenticated governs whether external senders can send to this mailbox. Selecting this option rejects messages from external senders even if they are defined in the list of allowed senders.

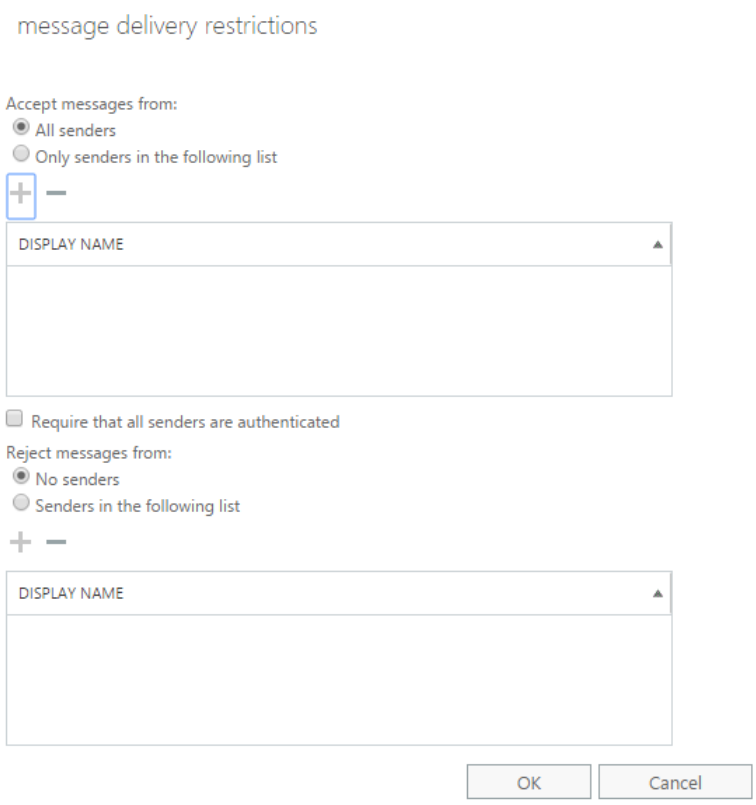

Figure 0-20: Configure message delivery restrictions

If you would rather block a list of people from sending to this mailbox select **Senders in the following list** under *Reject Messages From*. Click the **Add** button and select the people who you want to prevent from sending to this mailbox.

Whether you pick to build the allow list or the reject list depends on the scope of the block. If you want to reject most start building an allow list. If you want to accept most then start building a reject list.

Let's explore how to configure these options with the EMS.

To change the policies assigned to a mailbox we use the **Set-Mailbox** command. In the example below we are assigning a new retention policy to the user.

```
[PS] C:\> Set-Mailbox -Identity "Steve Smith" -RetentionPolicy "Delete items after 7 years"
```

Using this same method, we can also assign other policies to the mailbox, such as an organization sharing policy (**-SharingPolicy**), a user rights assignment (**-RoleAssignmentPolicy**) and an address book policy (**-AddressBookPolicy**). We then specify the name of the policy after the parameter. If the policy name contains spaces, then we will need to surround it in quotation marks.

To disable client connectivity to a user's mailbox we use the **Set-CASMailbox** command. For example, to disable POP3 access to a mailbox we would use the following command.

```
[PS] C:\> Set-CASMailbox -Identity "Steve Smith" -PopEnabled $false
```

To enable POP3 access switch the **-PopEnabled** parameter back to **$true**. Using this same method, we can also enable or disable any of the other client connection protocols. Be default these are all set to enabled.

-OWAforDevices specifies whether the user can connect with a mobile device using the Outlook app.

-ActiveSyncEnabled specifies whether the user can connect with a mobile device using ActiveSync.

-PopEnabled specifies whether the user can connect with a POP client.

-ImapEnabled specifies whether the user can connect with an IMAP client.

-OWAEnabled specifies whether the user can access their mailbox from Outlook on the Web.

-MAPIEnabled specifies whether the user can access their mailbox with the Outlook client.

To place a mailbox on litigation hold we would use the **Set-Mailbox** command. In this command, we specify a duration of unlimited. Unlimited means that the hold is indefinite and is only removed when the hold is manually removed from the mailbox.

```
[PS] C:\> Set-Mailbox -Identity "Steve Smith" -LitigationHoldDuration "Unlimited" -
LitigationHoldEnabled $true
```

We can specify a duration of the hold in the format of DD.HH:MM:SS. We can also specify a hold comment to display to the user and a hold URL that the user can visit to obtain more information. These are purely optional. In the example below we set a retention period of 365 days. We include a retention comment and a retention URL to the user.

```
[PS] C:\> Set-Mailbox -Identity "Steve Smith" -LitigationHoldEnabled $true -LitigationHoldDuration
"365.00:00:00" -RetentionUrl "https://internal.space-corp.net/wiki/litigation/faq.aspx" -
RetentionComment "You are being placed on litigation hold."
```

To configure email forwarding we use the **-ForwardingAddress** parameter. In this example, we forward Steve Smith's email to Kim Steele. We can also keep a copy of the message in Steve's mailbox by setting the **-DeliverToMailboxAndForward** parameter to **$true**. Setting this to **$false** or omitting the parameter will only maintain a copy of the message in the forwarding mailbox.

```
[PS] C:\> Set-Mailbox -Identity "Steve Smith" -ForwardingAddress "Kim Steele" -
DeliverToMailboxAndForward $true
```

To configure the max send and receive size for a mailbox, we use the **Set-Mailbox** command. Like how we set mailbox quotas in the prior section, specify a unit of measure in bytes (B), kilobytes (KB), megabytes (MB),

gigabytes (GB), or, terabytes (TB). Neglecting to add the unit of measure will set the value in bytes. In this example, we give Steve a max send and receive size of 10 megabytes.

```
[PS] C:\> Set-Mailbox -Identity "Steve Smith" -MaxReceiveSize 10MB -MaxSendSize 10MB
```

If we wanted to restrict who could send to Steve we could configure an allow or block list. To configure a list of allowed senders we would issue the following command.

```
[PS] C:\> Set-Mailbox -Identity "Steve Smith" -AcceptMessagesOnlyFrom @{Add="Joe Diaz","Ed
Patterson","Pam Kipling"}
```

To remove an allowed sender, we can switch the add statement to remove.

```
[PS] C:\> Set-Mailbox -Identity "Steve Smith" -AcceptMessagesOnlyFrom @{Remove="Joe Diaz"}
```

To only accept messages from authorized internal senders we use **-RequireSenderAuthenticationEnabled** and set it to a value of **$true**.

```
[PS] C:\> Set-Mailbox -Identity "Steve Smith" -RequireSenderAuthenticationEnabled $true
```

Alternatively, we can create a list of blocked senders. To do this we use the **-RejectMessagesFrom** parameter.

```
[PS] C:\> Set-Mailbox -Identity "Steve Smith" -RejectMessagesFrom @{Add="Kendal Ayers"}
```

Note: For an in-depth look at the **Set-CASMailbox** cmdlet check the following article https://technet.microsoft.com/en-us/library/bb125264(v=exchg.160).aspx

Mailbox delegation

Mailbox delegation allows an administrator to grant one user the ability to perform certain actions of another user. These delegations include Send As, Send on Behalf and Full Access.

Send As permissions grant the delegate the ability to send mail as the mailbox owner. Recipients of these messages will see the message as coming from the mailbox owner and not the delegate. This allows for impersonation of the mailbox owner. This permission may be best suited to an assistant who frequently transcribes emails. This permission only grants send rights. The delegate cannot view the contents of the mailbox.

Send on Behalf permissions also grant the delegate the ability to send mail as the mailbox owner. Recipients of these messages will see that the message has been sent by the delegate on behalf of the mailbox owner. This is suited for situations when a delegate needs send rights but more transparency or accountability needs to be established. This permission only grants send rights. The delegate cannot view the contents of the mailbox.

Full Access permissions allow the delegate full control over the mailbox. The delegate can perform many of the same tasks the mailbox owner can perform; including reading, editing and deleting objects and folders. Full access does not grant any send permissions. To send as the mailbox owner the delegate must also be assigned *Send As* or *Send on Behalf* permissions.

Let's explore assigning these permissions in both the EAC and EMS.

For these examples, we are going to assign Morgan Clarkson as a delegate of Harvey Lovell. We will allow Morgan full access rights to Harvey's mailbox and the ability to send messages as if they were coming from Harvey himself.

From the **Recipients** tab on the left select **Mailboxes** sub tab across the top. From here you can either double click the mailbox you want to edit, or, select the mailbox and click the **Edit** button (represented by a pencil).

This will launch the properties of the mailbox we want to assign a delegate. Select the **Mailbox Delegates** tab (Figure 6-21).

To assign the Send As permission click the **Add** button under the *Send As* section. Select the desired delegate from the list, click **Add** and then **Ok**. The user will be added as a delegate. To remove a delegate from Send As, select that user and click the **Remove** button. In our example, we have added Morgan Clarkson to this list.

To assign the Send on Behalf permission click the **Add** button under the *Send on Behalf* section. Select the desired delegate from the list, click **Add** and then **Ok**. The user will be added as a delegate. To remove a delegate from Send on Behalf, select that user and click the **Remove** button.

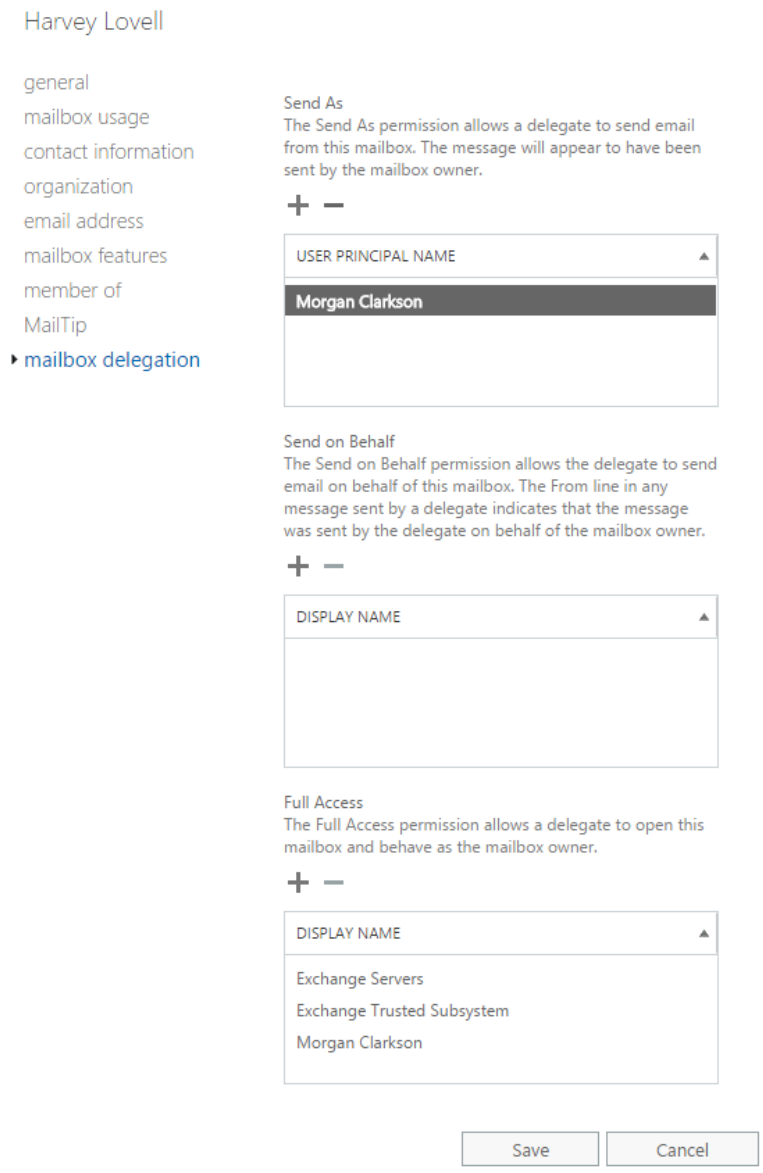

Figure 0-21: Assigning delegates to a mailbox (Send As, Send on Behalf, Full Access)

To assign the Full Access permission click the **Add** button under the *Full Access* section. Select the desired delegate from the list, click **Add** and then **Ok**. The user will be added as a full access delegate. To remove a delegate from Full Access, select that user and click the **Remove** button. In our example, we have added Morgan Clarkson to this list.

To assign Send As permissions with EMS we need to use the **Add-ADPermission** command.

```
[PS] C:\> Add-ADPermission -Identity "Harvey Lovell" -User "Morgan Clarkson" -ExtendedRights "Send As"
```

In this command **–User** specifies the delegate and **–Identity** specifies which mailbox they need to send from. Removal of the delegate is pretty much the same command. Except this time, we swap the **Add** verb for **Remove**. The rest remains the same.

```
[PS] C:\> Remove-ADPermission -Identity "Harvey Lovell" -User "Morgan Clarkson" -ExtendedRights "Send As"
```

To assign send on behalf permissions we use the **Set-Mailbox** command. The **–GrantSendOnBehalfTo** parameter is a multivalued field. This means it can contain multiple delegates. If we simply specify a delegate after this parameter, it will overwrite any existing delegates. To add an additional delegate, we use the following syntax.

```
[PS] C:\> Set-Mailbox -Identity "Harvey Lovell" -GrantSendOnBehalfTo @{Add="Morgan Clarkson"}
```

We can also add multiple delegates at the same time using comma separation. In the example below we assign three delegates to Harvey's mailbox.

```
[PS] C:\> Set-Mailbox -Identity "Harvey Lovell" -GrantSendOnBehalfTo @{Add="Rachel Hughes","Ed Patterson","Nick Haywood"}
```

We use similar syntax when we want to remove a delegate. Except this time, we specify **Remove**.

```
[PS] C:\> Set-Mailbox -Identity "Harvey Lovell" -GrantSendOnBehalfTo @{Remove="Rachel Hughes","Ed Patterson","Nick Haywood"}
```

To determine the current list of Send on Behalf delegates we can use the **Get-Mailbox** command.

```
[PS] C:\> Get-Mailbox -Identity "Harvey Lovell" | Format-List Name, GrantSendOnBehalfTo

Name                : Harvey Lovell
GrantSendOnBehalfTo : {space-corp.net/Employees/Production/Ed Patterson, space-
corp.net/Employees/Production/Nick Haywood, space-corp.net/Employees/R&D/Rachel Hughes}
```

To assign full access permissions we leverage the **Add-MailboxPermission** command.

```
[PS] C:\> Add-MailboxPermission -Identity "Harvey Lovell" -User "Morgan Clarkson" -AccessRights FullAccess
```

Like before we can remove the full access permission by flipping the **Add** verb to **Remove**.

```
[PS] C:\> Remove-MailboxPermission -Identity "Harvey Lovell" -User "Morgan Clarkson" -AccessRights FullAccess
```

To determine who has full access to Harvey's mailbox we issue the **Get-MailboxPermission** cmdlet. Running this command alone would generate a lot of extra information we do not need. For example, we would see inherited rights granted to the Exchange Trusted Subsystem or Harvey's own permissions to his mailbox. To eliminate this noise, we will filter our output using the **Where** statement.

```
[PS] C:\> Get-MailboxPermission -Identity "Harvey Lovell" | Where {$_.user.tostring() -ne "NT AUTHORITY\SELF" -and $_.IsInherited -eq $false} | Select Identity, User, {$_.AccessRights}

Identity                                          User                        $_.AccessRights
--------                                          ----                        ---------------
space-corp.net/Employees/Corporate/Harvey Lovell  SPACE-CORP\morgan.clarkson   FullAccess
```

The operator **–ne** refers to not equal. So, in the case of the filter above we are looking for rights not equal to NTAUTHORITY\SELF. More specifically we do not want the output to contain Harvey's permissions to his own mailbox. The **–eq** operator refers to equal. In the filter, we specify that inheritance must equal false. We tie this together with an **–and** operator which requires both conditions be matched.

AutoMapping

When adding a full access delegate through the EAC that mailbox is automatically added to the delegates Outlook profile the next time the Autodiscover process runs (or when the user closes and reopens Outlook). This feature is also known as AutoMapping. If a user does not want the mailbox to populate in their Outlook profile—perhaps they want to access the mailbox on demand—we would need to disable AutoMapping while assigning the delegation. This option is not available in the EAC so we will need to leverage EMS for this task. To do this we add the parameter **–AutoMapping** and set it to false.

```
[PS] C:\> Add-MailboxPermission –Identity "Harvey Lovell" -User "Morgan Clarkson" -AccessRights
FullAccess –AutoMapping $false
```

There is no way to remove the AutoMapping feature if you already have the delegation established. If this is the case, the delegation will need to be removed and recreated with the AutoMapping parameter.

Assigning folder permissions

There may be cases where granting a user full access over another mailbox is too lenient. The mailbox owner may prefer that the delegate only has access to certain folders. We can accomplish this goal through folder level permissions.

You can assign either individual access rights to a delegate, or, you can assign an access role. An access role is a grouping of common access rights. Table 16-1 lists the individual access rights and what permissions they grant.

Access Right	***Description***
ReadItems	Delegate can read all items within the designated folder
CreateItems	Delegate can create items within the designated folder
EditOwnedItems	Delegate can edit items they have created in the designated folder
DeleteOwnedItems	Delegate can delete items they have created in the designated folder
EditAllItems	Delegate can edit any item in the designated folder
DeleteAllItems	Delegate can delete any item in the designated folder
CreateSubfolders	Delegate can create subfolders in the designated folder
FolderOwner	Delegate can create subfolders in the designated folder. Delegate can also view and move the folder. Delegate cannot view, create, edit, or, delete any items in that folder.
FolderContact (public folders only)	Delegate is the contact for the public folder
FolderVisible (public folders only)	Delegate can view the public folder. Delegate cannot view or edit items in that public folder.
LimitedDetails (calendar only)	Delegate can view availability data including subject and location
AvailabilityOnly (calendar only)	Delegate can only view availability data

Table 0-1: Mailbox Folder Permissions Access Rights

Table 16-2 lists each access role, and which access rights they contain.

Access Role	***Access Rights***
Owner	ReadItems, CreateItems, EditOwnedItems, DeleteOwnedItems, EditAllItems, DeleteAllItems, CreateSubfolders, FolderOwner, FolderContact, FolderVisible.
PublishingEditor	ReadItems, CreateItems, EditOwnedItems, DeleteOwnedItems, EditAllItems, DeleteAllItems, CreateSubfolders, FolderVisible.
Editor	ReadItems, CreateItems, EditOwnedItems, DeleteOwnedItems, EditAllItems, DeleteAllItems, FolderVisible.

PublishingAuthor	ReadItems, CreateItems, EditOwnedItems, DeleteOwnedItems, CreateSubfolders, FolderVisible.
Author	ReadItems, CreateItems, EditOwnedItems, DeleteOwnedItems, FolderVisible.
NonEditingAuthor	ReadItems, CreateItems, FolderVisible
Reviewer	ReadItems, FolderVisible
Contributor	CreateItems, FolderVisible
None	FolderVisible

Table 0-2: Mailbox Folder Permissions Access Roles

Folder permissions can only be assigned using EMS. For this task, we utilize the **Add-MailboxFolderPermission** command. In this example, we grant the *Owner* access role to Rachel Hughes for Ed Patterson's inbox folder.

```
[PS] C:\> Add-MailboxFolderPermission -Identity "ed.patterson@space-corp.net:\Inbox" -User "Rachel
Hughes" -AccessRights Owner
```

To determine the existing folder permissions for Ed's inbox we can issue the **Get-MailboxFolderPermission** command.

```
[PS] C:\> Get-MailboxFolderPermission -Identity "Ed Patterson:\Inbox"

FolderName          User                AccessRights
----------          ----                ------------
Inbox               Default             {None}
Inbox               Anonymous           {None}
Inbox               Rachel Hughes       {Owner}
```

If we need to modify an existing permission, we can use the **Set-MailboxFolderPermission** command. In this example, we switch Rachel's folder permission from *Owner* to *Publishing Editor*.

```
[PS] C:\> Set-MailboxFolderPermission -Identity "ed.patterson@space-corp.net:\Inbox" -User "Rachel
Hughes" -AccessRights PublishingEditor
```

To revoke Rachel's folder permissions to Ed's inbox we issue the **Remove-MailboxFolderPermission** command.

```
[PS] C:\> Remove-MailboxFolderPermission -Identity "ed.patterson@space-corp.net:\Inbox" -User
"Rachel Hughes"

Confirm
Are you sure you want to perform this action?
Removing mailbox folder permission on "ed.patterson@space-corp.net:\Inbox" for user "Rachel Hughes".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [?] Help (default is "Y"):
```

Another possibility is when a delegate needs to be granted greater detail into a user's calendar. While the user can configure these options themselves through the calendar properties in Outlook it is also possible to do this with EMS. For example, let's assume Rachel needs to see the details of Ed's meetings. By default, Rachel can only see when Ed is either free or busy. To grant Rachel greater visibility we would use the *LimitedDetails* right as defined in Table 16-1. Our command would look like this.

```
[PS] C:\> Add-MailBoxFolderPermission "Ed Patterson:\Calendar" –User "Rachel Hughes" –AccessRights
LimitedDetails
```

If Ed wanted everyone to be able to see his meeting details he could modify the permissions assigned to the default user as detailed in the command below. If he ever needed to revoke this right, he can rerun the command with the **-AccessRights** parameter set to *AvailabilityOnly*.

```
[PS] C:\> Set-MailBoxFolderPermission "Ed Patterson:\Calendar" –User "Default" –AccessRights
LimitedDetails
```

Bulk manipulation of mailbox properties

While the bulk manipulation of mailbox properties can be performed via the EAC or EMS, it is much more efficient to do this via the command line.

To manipulate multiple objects via the EAC perform either a SHIFT-mouse-click to select a range of objects, or, CTRL-mouse-click to select specific objects. Once selected pick one of the items in the action pane under *Bulk Edit*. In Figure 6-22 we have selected three objects using the CTRL-mouse-click method. If we want to update the contact information for these mailboxes we can select the first *Update* link in the action pane.

To perform a similar task in EMS we would issue the following command. In this example, we retrieve all users in an organizational unit named *R&D*. We pipe the results from that command into the Set-User cmdlet.

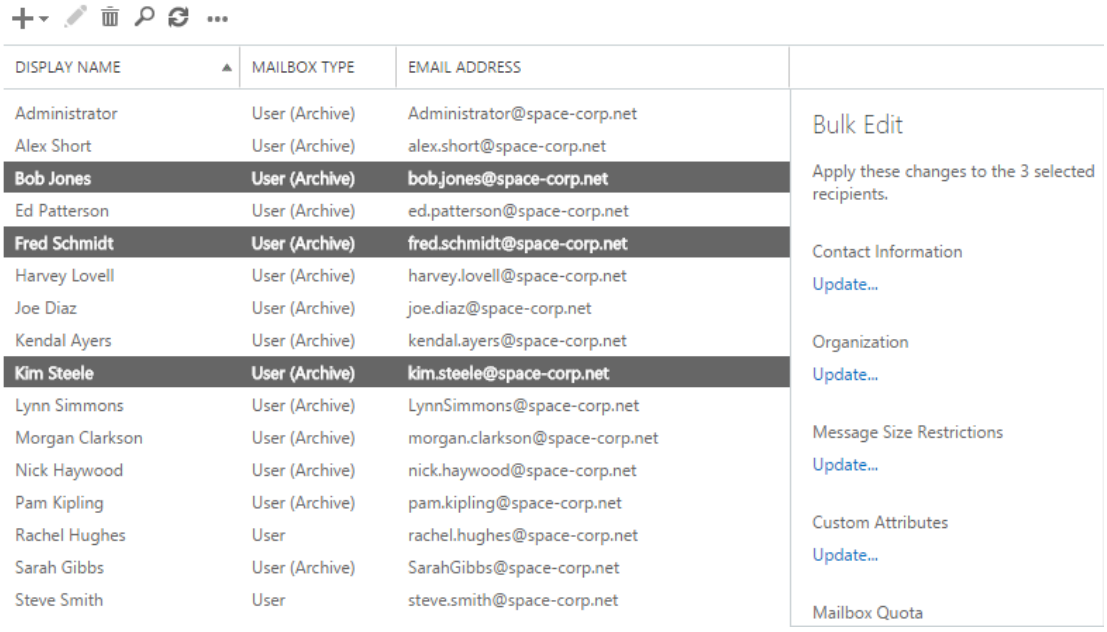

Figure 0-22: Bulk manipulation of mailbox objects

```
Get-User -OrganizationalUnit "space-corp.net/Employees/R&D" | Set-User -StreetAddress "400 Rocket Avenue Suite 101" -City "Merritt Island" -StateOrProvince "FL" -PostalCode "32953" -CountryOrRegion "US"
```

If you are unsure as to the implications of running a command you can attach the **-WhatIf** parameter. For the command above it will identify which user objects are to be modified. No objects are changed while this parameter is present. If you are satisfied with the result omit **-WhatIf** and rerun the command.

```
[PS] C:\> Get-User -OrganizationalUnit "space-corp.net/Employees/R&D" | Set-User -StreetAddress "400 Rocket Avenue Suite 101" -City "Merritt Island" -StateOrProvince "FL" -PostalCode "32953" -CountryOrRegion "US" -WhatIf
What if: Setting user "space-corp.net/Employees/R&D/Pam Kipling".
What if: Setting user "space-corp.net/Employees/R&D/Alex Short".
What if: Setting user "space-corp.net/Employees/R&D/Rachel Hughes".
```

We can also use filters to find users with a certain value in a field. In this example, we look for all users that have a company value that equals *'Space Corp'*. We then pipe those results into the Set-Mailbox cmdlet, where we configure a sharing policy.

```
[PS] C:\> Get-User -Filter {Company -eq "Space Corp"} | Set-Mailbox -SharingPolicy "Robotomics Sharing Policy"
```

If we wanted to disable POP services for each user that command might look like the following.

```
[PS] C:\> Get-CASMailbox | Set-CASMailbox -PopEnabled $false
```

Or if we wanted to make sure to only look for mailboxes where POP was enabled our command would change to the following.

```
[PS] C:\> Get-CASMailbox | Where-Object PopEnabled -eq $true | Set-CASMailbox -PopEnabled $false
```

Earlier in this chapter we explored using a filter when identifying which user mailboxes did not have an archive. Here is that example again.

```
[PS] C:\> Get-Mailbox -Filter {ArchiveState -eq "None" -and RecipientTypeDetails -eq "UserMailbox"}
| Enable-Mailbox -Archive -ArchiveDatabase DB02
```

In this filter, we looked for two conditions. First we required that the recipient type be a user mailbox. Second we required that the archive state on that user mailbox equalled none. If both conditions were true we enabled an archive for those users on DB02. The -and operator requires that both conditions be met. In the example below we use an **-or** operator to specify that only one of the conditions needs matched. Results from either condition are passed to Set-Mailbox which is next in the pipe.

```
[PS] C:\> Get-User -Filter {Department -eq "R&D" -or Department -eq "Sales"} | Set-Mailbox -
RetentionPolicy "Delete items after 7 years"
```

We can also query a distribution list for its members and pipe those results into a Set- verb. In this example, we ask Exchange to enumerate all members of a distribution group named *R&D*. Those members are then placed on an In-Place hold.

```
[PS] C:\> Get-DistributionGroupMember -Identity "R&D" | Set-Mailbox -LitigationHoldEnabled $true -
LitigationHoldDuration "365.00:00:00" -RetentionUrl "https://internal.space-
corp.net/wiki/litigation/faq.aspx" -RetentionComment "You are being placed on litigation hold."
```

To bulk apply delegate rights we could follow a similar process. In this example, we assign members of the customer service distribution group send as rights to the customer service shared mailbox. We will need to process each result from the pipe into a **ForEach** loop. The **-User** parameter is updated each time the ForEach loop runs.

```
[PS] C:\> Get-DistributionGroupMember -Identity "Customer Service" | ForEach-Object {Add-
ADPermission -Identity "CS Mailbox" -ExtendedRights "Send As" -User $_.Name}

Identity                                User                        DenY   Inherited
--------                                ----                        ----   ---------
space-corp.net/Employees/Sales/CS Mailbox   SPACE-CORP\pam.kipling       False  False
space-corp.net/Employees/Sales/CS Mailbox   SPACE-CORP\rachel.hughes     False  False
space-corp.net/Employees/Sales/CS Mailbox   SPACE-CORP\joe.diaz          False  False
```

Another use case is if we need to move all users from one database to another. In this example, we get all mailboxes with a property of DB02. We then pipe those results into a move request which targets the mailboxes to DB01.

```
[PS] C:\> Get-Mailbox -Database DB02 | New-MoveRequest -TargetDatabase "DB01" -BadItemLimit 10

WARNING: When an item can't be read from the source database or it can't be written to the
destination database, it will be considered corrupted. By specifying a non-zero BadItemLimit, you
are requesting Exchange not copy such items to the destination mailbox. At move completion, these
corrupted items will not be available at the destination mailbox.

DisplayName          StatusDetail          TotalMailboxSize             PercentComplete
-----------          ------------          ----------------             ---------------
Sarah Gibbs          WaitingForJobPickup   87.86 KB (89,967 bytes)      0
Joe Diaz             WaitingForJobPickup   0 B (0 bytes)                0
Morgan Clarkson      WaitingForJobPickup   0 B (0 bytes)                0
CS Mailbox           WaitingForJobPickup   2.734 KB (2,800 bytes)       0
```

Retrieving and exporting mailbox statistics

A common task for an Exchange administrator is to examine mailbox size statistics. Mailbox statistics can be retrieved by EAC by going into properties of a mailbox and viewing the **Mailbox Usage** tab. However, this is cumbersome when gathering a large amount of mailbox size data. This task is much more efficient via EMS. To accomplish this task with EMS we need to leverage the **Get-MailboxStatistics** cmdlet.

```
[PS] C:\> Get-MailboxStatistics -Identity "Sarah Gibbs"

DisplayName           ItemCount            StorageLimitStatus           LastLogonTime
-----------           ---------            ------------------           -------------
Sarah Gibbs           12                                                12/31/2016 2:52:52 PM
```

If we want to get statistics for all our mailboxes we can pipe the **Get-Mailbox** cmdlet into **Get-MailboxStatistics**. We are also tweaking which fields are returned and presenting them in a table format.

```
[PS] C:\> Get-Mailbox | Get-MailboxStatistics | Format-Table DisplayName, ItemCount,
TotalItemSize, LastLogonTime -AutoSize

DisplayName    ItemCount      TotalItemSize                        LastLogonTime
-----------    ---------      -------------                        -------------
Administrator  42,112         111.7 MB (117,125,939 bytes)         1/16/2017 1:35:11 PM
Steve Smith    16.456         146.2 MB (153,301,811 bytes)         1/16/2017 1:36:14 PM
Kim Steele     4,789          105.9 MB (111,044,198 bytes)         1/16/2017 1:39:14 PM
Sarah Gibbs    12,029         120.6 MB (126,458,265 bytes)         12/31/2016 2:52:52 PM
Lynn Simmons   6,987          37.67 MB (39,499,857 bytes)          12/29/2016 1:30:45 AM
Bob Jones      3,213          57.07 MB (59,842,232 bytes)          1/18/2017 1:36:14 PM
Pam Kipling    1,456          6.767 MB (7,095,713 bytes)           12/15/2017 1:36:14 PM
Alex Short     30,112         57.47 MB (60,261,662 bytes)          1/16/2017 3:46:14 PM
Rachel Hughes  9,984          63.51 MB (66,595,061 bytes)          1/18/2017 4:20:08 PM
Kendal Ayers   30,456         56.86 MB (59,622,031 bytes)          1/18/2017 6:23:18 PM
Ed Patterson   4,024          21.3 MB (22,334,668 bytes)           1/20/2017 4:19:31 PM
Joe Diaz       3,099          63.73 MB (66,825,748 bytes)          1/23/2017 5:49:56 PM
Fred Schmidt   3,926          65.99 MB (69,195,530 bytes)          1/23/2017 5:32:22 PM
```

If you need to standardize the unit of measure under the TotalItemSize column we can modify our command to include *Value.ToMB()* instruction. Note that this could be any unit of measure including *Value.ToB* for bytes, *Value.ToKB* for kilobytes and *Value.ToGB* for gigabytes.

We will also rename that column header with the **Label** syntax. Otherwise the TotalItemSize column will be renamed *"@{Expression={$_.TotalItemSize.Value.ToMB()};"*. We also sort the output in descending order with the **Sort-Object** cmdlet.

```
[PS] C:\> Get-Mailbox | Get-MailboxStatistics | Sort-Object TotalItemSize -Descending | Format-Table
DisplayName, ItemCount, @{Expression={$_.TotalItemSize.Value.ToMB()}; Label="Mailbox Size in MB"},
LastLogonTime -AutoSize

DisplayName    ItemCount      Mailbox Size in MB                   LastLogonTime
-----------    ---------      ------------------                   -------------
Sarah Gibbs    12,029         146.2 MB (153,301,811 bytes)         12/31/2016 14:52
Kim Steele     4,789          120.6 MB (126,458,265 bytes)         1/16/2017 13:39
Steve Smith    16.456         111.7 MB (117,125,939 bytes)         1/16/2017 13:36
Administrator  42,112         105.9 MB (111,044,198 bytes)         1/16/2017 13:35
Fred Schmidt   3,926          65.99 MB (69,195,530 bytes)          1/23/2017 17:32
Joe Diaz       3,099          63.73 MB (66,825,748 bytes)          1/23/2017 17:49
Ed Patterson   4,024          63.51 MB (66,595,061 bytes)          1/20/2017 16:19
Rachel Hughes  9,984          57.47 MB (60,261,662 bytes)          1/18/2017 16:20
Alex Short     30,112         57.07 MB (59,842,232 bytes)          1/16/2017 15:46
Pam Kipling    1,456          56.86 MB (59,622,031 bytes)          12/15/2017 13:36
Bob Jones      3,213          37.67 MB (39,499,857 bytes)          1/18/2017 13:36
Lynn Simmons   6,987          21.3 MB (22,334,668 bytes)           12/29/2016 1:30
Kendal Ayers   30,456         6.767 MB (7,095,713 bytes)           1/18/2017 18:23
```

To export this output to a CSV file instead of displaying on the console we can use the **Export-CSV** cmdlet. Following this command, we specify the path to the CSV file. If your file path contains spaces, you will need to enclose it in quotation marks. The **-NoTypeInformation** parameter removes the type information normally displayed in the first row of the export file.

```
[PS] C:\> Get-Mailbox | Get-MailboxStatistics | Sort-Object TotalItemSize -Descending | Select
DisplayName, ItemCount, @{Expression={$_.TotalItemSize.Value.ToMB()}; Label="Mailbox Size in MB"},
LastLogonTime | Export-CSV C:\Users\Administrator\Desktop\MailboxStatistics.csv -NoTypeInformation
```

Part of the information displayed in the default output is the LastLogonTime. This tells us when the mailbox was last logged onto. Another great field is the LastLoggedOnUserAccount. Using this information, we could look for mailboxes that have not been accessed in more than x days. In our example, below we look for user mailboxes that have not been accessed in over 90 days.

```
[PS] C:\> Get-Mailbox -Filter {RecipientTypeDetails -eq "UserMailbox"} | Get-MailboxStatistics |
Where {$_.LastLogonTime -lt (Get-Date).AddDays(-90)} | Select DisplayName, LastlogonTime,
LastLoggedOnUserAccount | Export-CSV C:\Users\Administrator\Desktop\LogonsOver90Days.csv -
NoTypeInformation
```

Note: For an in-depth look at the **Get-MailboxStatistics** cmdlet check the following article https://technet.microsoft.com/en-us/library/bb124612%28v=exchg.160%29.aspx

Shared Mailboxes

A shared mailbox is a mailbox specifically designed for the purposes of collaboration. To grant users access to a shared mailbox they must be delegated the full access permission. Likewise, to send as a shared mailbox those users must be delegated send as rights. You may wonder how this differs from a standard user mailbox. You can after all delegate these same rights to a user mailbox.

The subtle difference is that when a shared mailbox is created in Active Directory its account is disabled. Unlike a user mailbox a shared mailbox cannot be directly logged into. Instead it must be attached as a secondary mailbox in Outlook or Outlook on the Web. The benefit here is that a shared mailbox does not consume a client access license (CAL) which translates to cost savings for the business.

Business use cases for shared mailboxes could include a customer service mailbox, where multiple customer service agents have their own primary mailboxes but also need to manage service inquiries through a central location. This shared mailbox not only accounts for all service inquiries that have been received but it also helps track which have been read and responded to.

Similar business use cases could include shared mailboxes for technical support. While it certainly doesn't replace a ticketing system it may be perfect for a smaller internal IT department with just a few representatives. Other alternatives could be accounts receivable, or, human resources.

In the following sections, we will explore how to create and manage shared mailboxes. Plus, we will highlight any subtle difference between their management and that of a standard user mailbox.

Creating a shared mailbox

In this section, we will look at how to create a shared mailbox. We will explore how to accomplish this task with both the EAC and EMS.

In this example, we are going to create a shared mailbox called Customer Service.

Select the **Recipients** tab on the left and **Shared** sub tab across the top. This tab lists all shared mailboxes in the environment. From here click the **New** button (represented by a plus sign).

On the *New Shared Mailbox* window (Figure 6-23) type a **Display name** and **Alias** for the mailbox. In our example, we have typed *CustomerService* for both fields.

Click **Browse** in the Organizational Unit section. This brings up the *Select an Organization Unit* dialog. From here you can select which Organization Unit (OU) you want the shared mailbox's user account to be created under.

Clicking the **Add** button under the *Users* section allows you to assign both full access and send as permissions to the specified user. You can remove a user by selecting that user from the list and clicking the **Remove** button. If you need to assign your users only full access or only send as permissions, then leave this list blank. Individual delegations can be assigned once the mailbox has been created. Clicking the **More options** link allows for additional items to be configured such as which database will host the shared mailbox.

One of the things you will notice is the absence of user logon and password fields. As mentioned in the prior section shared mailboxes are created as disabled accounts and not designed to be logged into. The name of the disabled account that is created in Active Directory will match the display name of the mailbox.

Click **Save**.

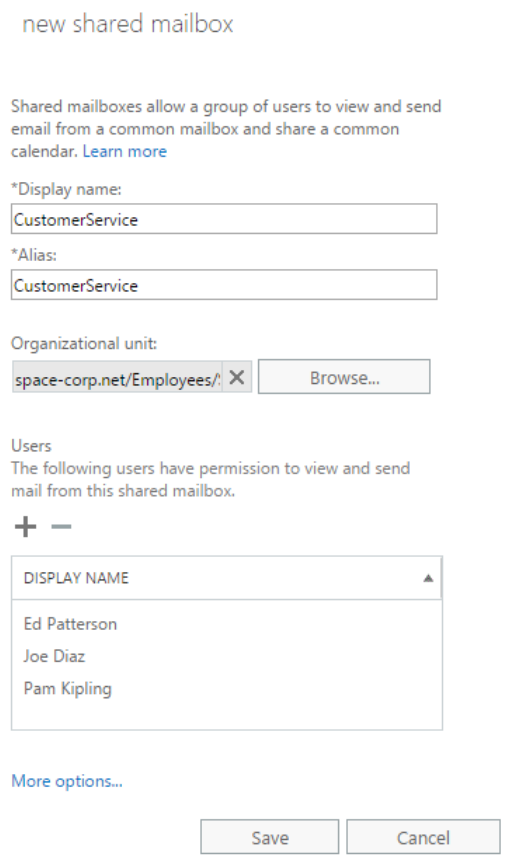

Figure 0-23: Creating a shared mailbox

Let's explore how we would have completed the same task but using the EMS. To do this we will use the **New-Mailbox** cmdlet which is the same cmdlet we use when creating a user mailbox. The main difference is the addition of a **-Shared** parameter. Using the above example of Customer Service let's see what the process would have looked like in PowerShell.

```
[PS] C:\> New-Mailbox -Shared -Name "CustomerService" -DisplayName "CustomerService" -Alias
"CustomerService" -OrganizationalUnit "space-corp.net/Employees/Sales"
```

Using the **-Shared** parameter instructs Exchange to create the mailbox as shared with a disabled account. This also allows the command to run without the user principle name and password parameters, which are normally required with this cmdlet.

From Figure 6-23 we also specified that three users should be granted full and send rights to the shared mailbox. To accomplish this EAC is running several more commands behind the scenes. To completely replicate this behaviour we would also need to run additional delegation commands. These are the same commands we would use when delegating access to a user mailbox.

```
Add-MailboxPermission -Identity "CustomerService" -User "Ed Patterson" -AccessRights "FullAccess"
Add-MailboxPermission -Identity "CustomerService" -User "Joe Diaz" -AccessRights "FullAccess"
Add-MailboxPermission -Identity "CustomerService" -User "Pam Kipling" -AccessRights "FullAccess"
Add-ADPermission -Identity "CustomerService" -User "Ed Patterson" -ExtendedRights "Send As"
Add-ADPermission -Identity "CustomerService" -User "Joe Diaz" -ExtendedRights "Send As"
Add-ADPermission -Identity "CustomerService" -User "Pam Kipling" -ExtendedRights "Send As"
```

Managing shared mailbox properties

With our shared mailbox created let's look at managing that mailbox through the EAC and EMS.

To access the properties of a shared mailbox, select the **Recipients** tab on the left and **Shared** sub tab across the top. From here you can either double click the mailbox you want to edit, or, select the mailbox and click the **Edit** button (represented by a pencil).

This will launch a dialog with several tabs. Most these tabs are identical to their user mailbox counterparts which we covered in the previous section titled *Managing Mailbox Properties*.

However, some of the tabs do possess subtle differences. One of those is the **General** tab. From here we can change the display name and alias and decide whether to hide this shared mailbox from all address lists. Unlike its user mailbox counterpart, the general tab contains no user account details.

The next tab is **Mailbox Delegation** which has been moved up the list of tabs for shared mailboxes. Like the user mailbox this tab allows an administrator to grant users the ability to perform certain actions against the shared mailbox. These delegations include Send As and Full Access. However, Send on Behalf has been removed.

Last up is the **Mailbox Features** tab. The big change is the absence of all phone and voice features. This eliminates unified messaging functionality for the shared mailbox and any mobile device features. Aside from these missing pieces all other settings are present.

When configuring these properties through EMS we use the same cmdlets as a user mailbox. For example, assigning full access and send as permissions follow the same syntax as their user mailbox counterparts. In this example, we grant user Joe Diaz full access and send as rights to the customer service shared mailbox.

```
[PS] C:\> Add-ADPermission -Identity "CustomerService" -User "Joe Diaz" -ExtendedRights "Send As"
[PS] C:\> Add-MailboxPermission -Identity "CustomerService" -User "Joe Diaz" -AccessRights
FullAccess
```

Like a user mailbox we can also set a retention policy on a shared mailbox, or, configure a In-Place hold. In the example below we assign a new retention policy to the shared mailbox using the **Set-Mailbox** cmdlet.

```
[PS] C:\> Set-Mailbox -Identity "CustomerService" -RetentionPolicy "Delete items after 7 years"
```

In this command, we place the customer service mailbox on In-Place hold.

```
[PS] C:\> Set-Mailbox -Identity "CustomerService" -LitigationHoldEnabled $true -
LitigationHoldDuration "365.00:00:00" -RetentionUrl "https://internal.space-
corp.net/wiki/litigation/faq.aspx" -RetentionComment "This mailbox has been placed on litigation
hold."
```

Resource Mailboxes

Resource mailboxes come in two flavours. A room mailbox and an equipment mailbox.

A room mailbox is a special type of resource mailbox that is designed to represent a physical meeting location. This location could be a conference room, training room, or, any other kind of room that requires scheduling. The typical business case for a room mailbox is to manage a conference room's availability for meetings and to avoid scheduling conflicts. But the room mailbox could be used for any number of meeting locations such as reserving a company basketball court or on-campus gym.

An equipment mailbox is very similar to a room mailbox. However, rather than identifying a physical meeting location an equipment mailbox identifies items that needs to be reserved. Examples of this could be a portable projector, a company truck, or, perhaps even an elliptical in the company gym. Equipment mailboxes may also have industry specific purposes. For example, a construction company might use equipment mailboxes to reserve construction equipment like excavators or bulldozers.

It might seem that the distinction between room and equipment mailboxes is purely aesthetics. In many cases, they are identical. The distinction is that an equipment mailbox will not show up in the Outlook Room Finder as a place to meet.

Resource mailboxes can be configured to automatically process meeting requests or be subject to booking approval by a designated user. Like a shared mailbox resource mailboxes are created with a disabled user account. This means that room and equipment mailboxes do not consume client access license. In the following sections, we explore how to create and manage resource mailboxes through the EAC and EMS.

Creating a room mailbox

In the following example, we are going to create a new room mailbox for our executive conference room.

Select the **Recipients** tab on the left and **Resources** sub tab across the top. From here click the **New** button (represented by a plus sign) and select **Room Mailbox**.

On the *New Room Mailbox* window (Figure 6-24) type a room name and an alias. In our example, we have typed *Executive Conference Room* for the name and *ExecConf* for the alias.

Click **Browse** under the *Organizational Unit* section to specify which Organization Unit (OU) you want the disabled account for the room mailbox to be created under. Leaving this blank will create the account under the default users OU.

Enter a value into the **Location**, **Phone** and **Capacity** fields. These fields are optional so they can be kept blank. Although these fields provide invaluable information to the user so it is recommended to fill them out. Clicking the **More options** link allows for additional items to be configured such as which database will host the room mailbox.

Click **Save**. Once back on the resources tab you will see a new mailbox with a mailbox type of room.

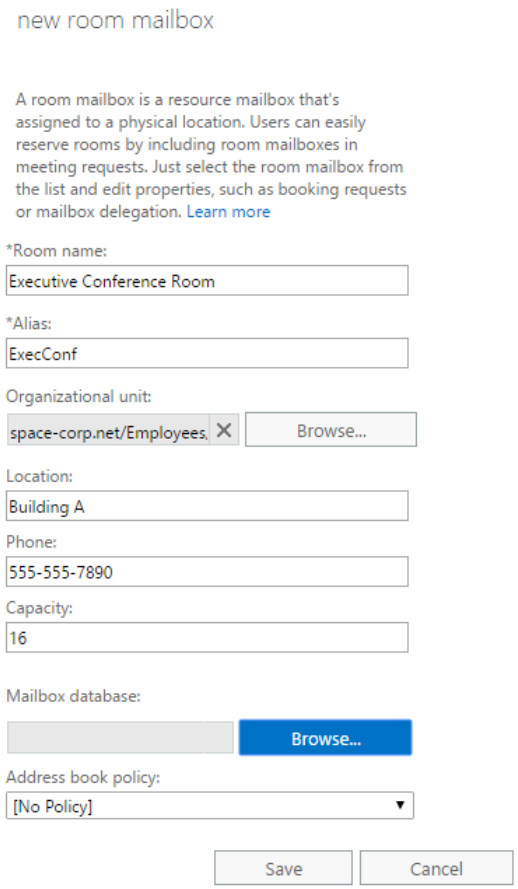

Figure 0-24: New Room Mailbox

To complete this same task in the EMS we actually need to run up to three commands. The first is the **New-Mailbox** cmdlet in combination with the **-Room** parameter.

```
[PS] C:\> New-Mailbox -Room -Name "Executive Conference Room" -DisplayName "Executive Conference Room" -Alias "ExecConf" -OrganizationalUnit "space-corp.net/Employees/Corporate" -ResourceCapacity "16"
```

The **-Room** parameter designates this as a room mailbox. This instructs Exchange to create a disabled account in the organizational unit specified. The name of the disabled account will equal the value specified in the **-Name** parameter. The **-ResourceCapacity** parameter matches the capacity field in the EAC. Inclusion of this parameter is optional.

To specify the location and phone fields we need to use the **Set-User** cmdlet. These are optional values so you can skip this command altogether if desired. The example, below illustrates how to configure the location and phone fields for our mailbox.

```
[PS] C:\> Set-User -Identity "Executive Conference Room" -Office "Building A" -Phone "555-555-8790"
```

The third and final command is to set the booking policy. By default, when the EAC creates a new room mailbox it sets the booking policy to automatically accept all meetings requests as long as there is no conflict or violation of the booking policy. Through EMS we do this by setting the **-AutomateProcessing** parameter to a value of **AutoAccept**. In addition, the EAC also sets additional booking parameters. We will cover these in a later section. For now, here is the command to replicate the behaviour of the EAC.

```
[PS] C:\> Set-CalendarProcessing -Identity "Executive Conference Room" -AutomateProcessing "AutoAccept" -AllBookInPolicy $true -AllRequestInPolicy $false
```

Creating a room list

A room list is a great way of grouping similar room mailboxes into a logical container. For example, if a company has multiple buildings or locations it might make sense to group those rooms by building or location. For a much larger company each room list could be designated by floor number. This allows users to more easily find rooms that are close in proximity to the attendees.

When using the scheduling assistant in Outlook on the Web (Figure 6-25) a user is presented with all available conference rooms for the time specified. Rather than search for a room a user can also select the option to find any room that is available in the room list. This is a good way for a user to quickly secure any available nearby room. The user can also choose an entirely different room list if the meeting is to be held elsewhere.

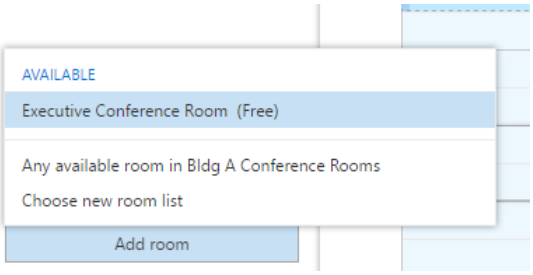

Figure 0-25: Using room lists in Outlook on the Web

There is no method for creating room lists in EAC, so let's explore this process in EMS. A room list is actually a modified distribution group so for this command we are going to utilize the **New-DistributionGroup** cmdlet.

```
[PS] C:\> New-DistributionGroup -Name "Bldg A Conference Rooms" -OrganizationalUnit "space-corp.net/Employees/Corporate" -RoomList
```

In this command, we apply the **-RoomList** parameter to identify that is the purpose of this distribution group. We also specify a name for the room list and which organizational unit where we want to place the object.

Next we need to add the room mailboxes as members of this distribution group. For this we use **Add-DistributionGroupMember** cmdlet. In this example, we add our *Executive Conference Room* to the new room list.

```
[PS] C:\> Add-DistributionGroupMember -Identity "Bldg A Conference Rooms" -Member "Executive Conference Room"
```

Creating an equipment mailbox

In the following example, we are going to create a new equipment mailbox for a portable projector.

From the **Recipients** tab on the left select the **Resources** sub tab across the top. From here click the **New** button and select **Equipment Mailbox**.

On the *New Equipment Mailbox* window (Figure 6-26) type an equipment name and an alias. In our example, we have typed *HD 1080 Projector* for the name and *HDProj* for the alias.

Click **Browse** under the *Organizational Unit* section to specify which Organization Unit (OU) you want the disabled account for the equipment mailbox to be created under. Leaving this blank will create the account under the default users OU. Clicking the **More options** link allows for additional items to be configured such as which database will host the equipment mailbox. Click **Save**.

Once back on the resources tab, under the *Mailbox Type* column, you will see that equipment mailboxes are designated as *Equipment* whereas room mailboxes are designated as *Room*.

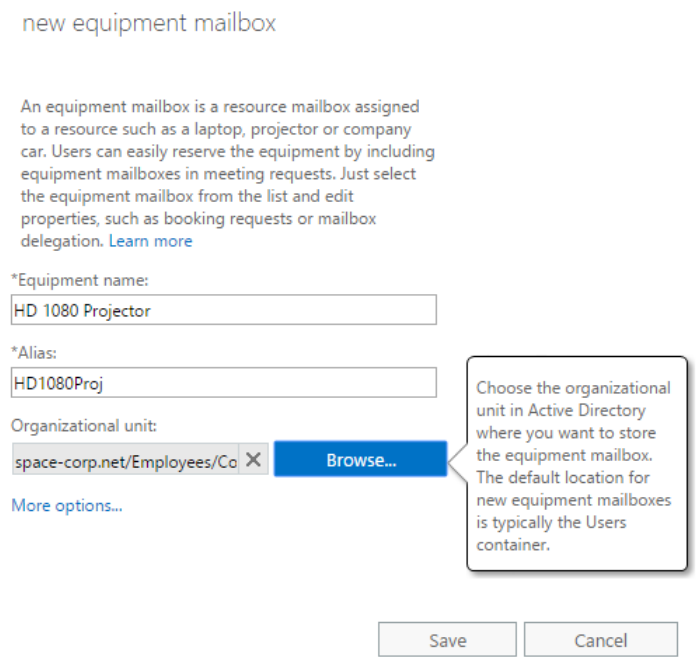

Figure 0-26: Creating an equipment mailbox

To perform this same action in EMS our command would be as follows.

```
[PS] C:\> New-Mailbox -Equipment -Name "HD 1080 Projector" -DisplayName "HD 1080 Projector" -Alias
"HD1080Proj" -OrganizationalUnit "space-corp.net/Employees/Corporate"
```

The difference between this command and when we created the room mailbox is the substitution of the **-Room** parameter for **-Equipment**. Unlike the room command we do not specify a resource capacity.

Configure booking options for rooms and equipment

Now that we have our resource mailbox created let's explore the advanced booking and scheduling options available in the EAC and EMS.

From the **Recipients** tab on the left select the **Resources** sub tab across the top. This tab lists all room and equipment mailboxes. Double click either a room or equipment mailbox to edit its properties and navigate to the **Booking Delegates** tab (Figure 6-27). This tab lists two options.

This first is **Accept or decline booking requests automatically**. This option fully automates the scheduling of a resource mailbox's calendar. If the resource is available, and as long as it complies with the booking options, the mailbox accepts the meeting request. If there is a conflict, or the request does not comply with the settings in the booking options tab, the mailbox rejects the meeting request.

The second is **Select delegates who can accept or decline booking requests**. This option activates the *Delegates* box immediately below where we can identify a list of users who either approve or decline meeting requests to that resource. This is a great way to moderate who can book a resource. For example, a business use case might be to restrict who can reserve an executive board room. To add a user, click the **Add** button. To remove a user, select that individual and click the **Remove** button.

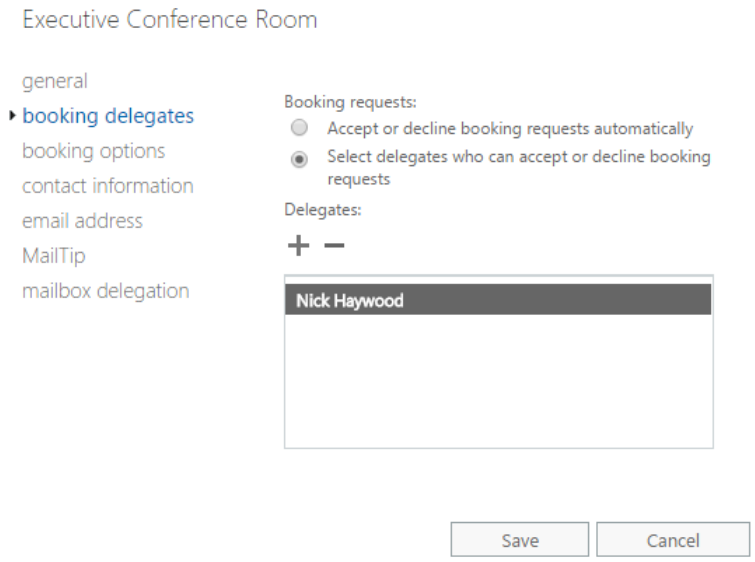

Figure 0-27: Booking Delegates

In our example in Figure 6-27 we have specified that Nick Haywood should process all meeting requests for the Executive Conference Room. Nick will receive an email each time this room receives a meeting request. Nick will have the option to either accept or decline the request.

One final option that is not identified in the Figure 6-27 is **Use customized setting to accept or decline booking requests**. If you have configured a booking option that is not configurable in the EAC, and is only available in EMS, this third option will appear.

The **Booking Options** tab (Figure 6-28) allows us to specify the criteria that a meeting request must adhere to. When a meeting adheres to these options it is considered in-policy. When a meeting request falls outside these thresholds it is considered out-of-policy. It is worth noting that some policy settings can only be set in EMS. We will cover a few of these options later in this section.

The first option **Allow repeating meetings** dictates whether recurring meetings can be scheduled against the resource. If unchecked only single instance meetings can be scheduled.

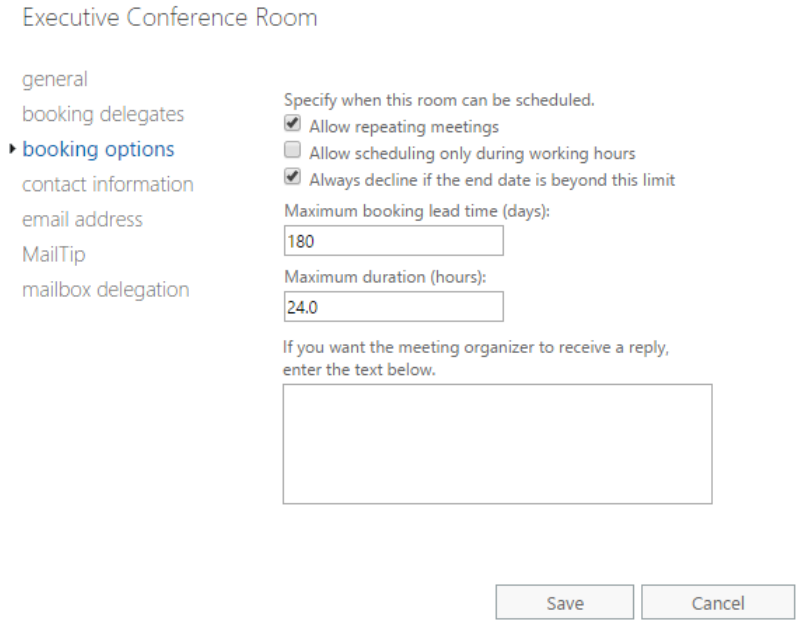

Figure 0-28: Booking Options

The next option **Allow scheduling only during working hours** determines whether meetings can be scheduled outside of working hours. Working hours are determined by the resource's own calendar settings. By default, the resource's calendar is set with working hours of Monday through Friday from 8am to 5pm. You can change a resource's working calendar by logging into that mailbox's Exchange Control Panel (ECP). To do this you would select your username in the top right corner of the EAC screen. From the drop down select **Another User**, pick the resource mailbox and click **Ok**. Next click the **Settings** tab and **Calendar** subtab. From here you can configure the work week and set the working hours. Click **Save**.

We can also configure working hours with EMS. To do this we use the **Set-MailboxCalendarConfiguration** cmdlet. In this example, we configure the Executive Conference Room with a start time of 7am and an end time of 7pm. The time is depicted using a 24-hour clock using the HH:MM:SS format. We also specify the room's time zone with the **-WorkingHoursTimeZone** parameter.

```
[PS] C:\> Set-MailboxCalendarConfiguration "Executive Conference Room" -WorkingHoursStartTime
07:00:00 -WorkingHoursEndTime 19:00:00 -Workdays AllDays -WorkingHoursTimeZone "Eastern Standard
Time"
```

The third option **Always decline if the end date is beyond this limit** will decline a meeting if it is scheduled beyond the value specified in **Maximum booking lead time**. The default value for maximum lead time is 180 days. Any meeting scheduled more than 180 days will be considered out-of-policy and will be declined. Setting a value of zero allows a meeting to be booked any number of days in advance.

The **Maximum Duration** field determines how long the meeting itself can be. The default value is 24 hours. Any meeting submitted that is longer than 24 hours will be considered out-of-policy and will be rejected. Setting a value of zero allows for an unlimited meeting length.

The final box allows us to specify a custom reply to send back to the meeting organizer.

For more granularity on booking permissions we can use the resources' control panel as mentioned previously. Many of those options can be found under the **Settings > Resources** tab. Alternatively, we can also use EMS. For example, rather than have a delegate assigned to a meeting room we could create a list of users or groups who are allowed to book the room. Anyone outside of this list would be rejected. In this command, we specify that only meeting requests from Harvey Lovell and Morgan Clarkson are allowed. We also configure the **-AllBookInPolicy** parameter to false. Leaving this parameter at its default value of true would cause calendar processing to ignore any user we set with the **-BookInPolicy** parameter.

```
[PS] C:\> Set-CalendarProcessing "Executive Conference Room" -AutomateProcessing AutoAccept -
AllBookInPolicy $false -BookInPolicy "Harvey Lovell","Morgan Clarkson"
```

Another example could be if we want to exclude a user from the booking policy. For example, the maximum default duration for a meeting is 24 hours. With this policy in affect anyone submitting a meeting that exceeds 24 hours is automatically declined as they are considered out-of-policy. If we wanted to allow a user or group to submit requests outside of this policy, we can use the **-RequestOutOfPolicy** parameter. For example, to allow Harvey Lovell to submit out-of-policy requests that would be accepted our command would look like this. All other users are still required to submit in-policy requests.

```
[PS] C:\> Set-CalendarProcessing "Executive Conference Room" -AutomateProcessing AutoAccept -
RequestOutOfPolicy "Harvey Lovell" -ResourceDelegates "Nick Haywood"
```

We can also use this command to decide whether external senders can book meetings to the resource. The default for this parameter is false.

```
[PS] C:\> Set-CalendarProcessing "Executive Conference Room" -ProcessExternalMeetingMessages $true
```

Note: For more information on booking options check this article https://technet.microsoft.com/en-US/library/ms.exch.eac.EditRoomMailbox_ResourceDelegates(EXCHG.160).aspx

AutoUpdate versus AutoAccept: In many of these examples we have used the parameter -AutomateProcessing. The purpose of this parameter is to determine whether or not a mailbox should automatically process meeting requests against a booking policy (AutoAccept), or, require the mailbox owner to manually accept or decline the request (AutoUpdate). By default, all resource mailboxes on creation are set to AutoAccept. In contrast, all user mailboxes are configured to AutoUpdate. It is possible to change the default behaviour of a resource mailbox from Autoaccept to Autoupdate. However, this would align the booking experience to that of any other meeting participant who needs to manually accept or decline.

Converting a mailbox type

It is possible to convert a mailbox from one type to another. One possible business case for converting a user mailbox to a shared mailbox could be part of an employee separation process. An ex-employee's email may need to be kept for continuity. Converting the mailbox to shared is a great way to deactivate the user account, so it cannot be logged into, while still being able to access the mailbox. It is also a great way to free up a client access license (CAL) for a user who is no longer with the company. This process cannot be completed in the EAC so let's jump straight into EMS.

To convert a mailbox from user to shared we use the **Set-Mailbox** cmdlet. In this example, we will convert Sarah Gibb's mailbox to a shared mailbox. We use the **-Type** parameter to perform the actual switch.

```
[PS] C:\> Set-Mailbox -Identity "Sarah Gibbs" -Type Shared
```

To confirm this change completed successfully we can run the following command.

```
[PS] C:\> Get-Mailbox -Identity "Sarah Gibbs" | Format-List Name, RecipientTypeDetails

Name                : Sarah Gibbs
RecipientTypeDetails : SharedMailbox
```

There may also be business cases where a shared mailbox needs to be converted back to a user mailbox. One example, could be that a shared role has expanded and grown into a full-time position requiring the mailbox be assigned to the new employee. To convert a shared mailbox into a user mailbox, issue the following command.

```
[PS] C:\> Set-Mailbox -Identity "Building Maintenance" -Type Regular
```

It is also possible to convert mailboxes to other types including resource mailboxes. There corresponding type values are **Room** and **Equipment**.

Reconnecting non-user mailboxes

The shared and resource tabs do not contain the option to reconnect a mailbox. To reconnect a shared or resource mailbox in the EAC you will need to navigate to the **Mailboxes** tab. From here select the **More** button (represented by an ellipsis) and select **Connect a Mailbox**.

An *Information* dialog will appear asking if you want to reattach the mailbox to the original AD account, or, a different account (Figure 6-8). If you reconnect the mailbox back to the original user account, the process will take a couple of seconds and put you back to the mailboxes tab. Navigate over to the shared or resource tabs and hit the **Refresh** button. The mailbox will reappear.

If we choose to reconnect to another user, the wizard continues. First it asks whether you want to alter the type of mailbox during reconnection. We can keep the mailbox in its original form, or, convert to a different type. For example, what was originally a shared mailbox could be transformed to a user mailbox. Once you have confirmed the type of mailbox click **Next** and specify which account to reattach the mailbox to. The remainder of the process is identical to that documented in the section titled *Reconnecting a disabled mailbox*.

To reconnect a mailbox with EMS is somewhat similar. Like a user mailbox we use the **Connect-Mailbox** cmdlet. The difference for a non-user mailbox is that we must specify an additional parameter. The **-Shared** parameter reattaches the mailbox as shared. Likewise, to reconnect a resource mailbox we would use either the **-Room** and **-Equipment** parameters. Omitting these parameters will reconnect the mailbox as a regular user. To reconnect a shared mailbox, issue the following command.

```
[PS] C:\> Connect-Mailbox -Identity "CustomerService" -Database "DB02" -User
"NewCustomerServiceAccount" -Shared
```

Linked Mailboxes

A linked mailbox is a mailbox that is accessed by a user from another forest. The forest where the mailbox is located is referred to the resource forest. The forest where the user is located is referred to the accounts forest. For this process to work a trust must exist between the two forests. This trust can be a one-way trust or a two-way trust. If a one-way trust the resource forest must trust the user forest.

A common business case for linked mailboxes could be the result of a merger or acquisition. Where both companies are consolidating mail services into a single forest while still maintaining security boundaries by having two separate forests.

Creating a linked mailbox

Prior to creating the linked mailbox, the user account in the accounts forest must already exist. The linked mailbox creation process does not create this account. The process does create a disabled account in the resource forest to link the mailbox.

From the **Recipients** tab on the left select the **Mailboxes** tab across the top. From here click the **New** button and select **Linked Mailbox**.

On the *New Linked Mailbox* window select the accounts forest that needs a mailbox. The number of entries in the drop-down is determined by the number of trusts you have with other forests. In our example, we select SKARO.LOCAL who we have a two-way forest trust established. Enter administrative credentials if prompted.

On the *Select Linked Master Account* page select a domain controller from the accounts forest. Exchange uses this domain controller to retrieve a list of user accounts. The *Linked Master Account* field is the user who needs the mailbox. Click **Browse** and select a user. In figure 6-29, we select user River Song from the SKARO forest. Click **Next**.

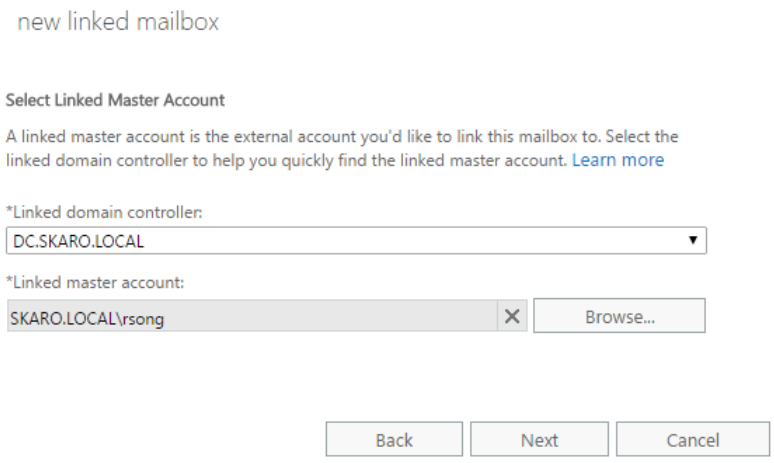

Figure 0-29: Creating a linked mailbox

On the *Enter General Information* page specify a name, organizational unit and logon name for the disabled account. The organizational unit will be the location of the disabled account in the resource forest. The logon information is not used by the user. It is a required field to get the disabled account created and is leveraged to generate the email address for the mailbox. Instead the user will continue to use credentials from their own forest which will be validated via the trust.

Selecting **More options** allows us to configure additional fields for the disabled account such as first name, last name, alias and what database should host the mailbox. If the alias is populated Exchange will use that value instead of the logon name to create the email address for the mailbox. Click **Finish**.

Back on the **Mailboxes** tab we will see our new mailbox with a *Mailbox Type* of *Linked*.

To perform this same task in EMS we run this command in the resource forest.

```
[PS] C:\> New-Mailbox -Name "River Song" -LinkedDomainController "DC" -LinkedMasterAccount
"SKARO\rsong" -OrganizationalUnit "space-corp.net/Employees/Corporate" -UserPrincipalName
river.song@space-corp.net -LinkedCredential:(Get-Credential SKARO\administrator)
```

In this command:

-LinkedDomainController specifies a domain controller in the accounts forest.

-LinkedMasterAccount identifies the user from the accounts forest that requires a mailbox.

-LinkedCredential instructs EMS to prompt for administrative credentials from the accounts forest.

-UserPrincipleName specifies the logon name for the disabled account in the resources forest. This is used to generate the email address of the mailbox unless the alias parameter is also specified.

-OrganizationalUnit determines the location of the disabled account in the resource forest.

Site Mailboxes

Site mailboxes were a concept first introduced in Exchange 2013 and carried over into Exchange 2016. In essence a site mailbox combines a document library from SharePoint and a mailbox from Exchange. It then allows these resources to be accessed through a common client such as Outlook. The benefit here is that what was originally two separate work streams for users has now been combined into a single more productive and more collaborative effort. The creation and management of site mailboxes are managed solely through SharePoint which is outside the scope of this book. However, some features can be managed through the Exchange Management Shell such as configuring a site mailbox provisioning policy.

A provisioning policy governs thresholds and naming prefixes for all site mailboxes. While multiple provisioning policies can exist in an environment only one policy can be in effect at any one time. This means that all site mailboxes are subject to whichever policy is designated as default. To create a new site mailbox policy, we issue the **New-SiteMailboxProvisioningPolicy** cmdlet. In this command, we have also specified mailbox quotas and send limits for all site mailboxes. The **-IsDefault** parameter also designates this new policy as the default.

```
[PS] C:\> New-SiteMailboxProvisioningPolicy -Name "SiteMailboxPolicy" -IsDefault -IssueWarningQuota
13GB -ProhibitSendReceiveQuota 15GB -MaxReceiveSize 100MB
```

If we wanted to alter the options of an existing policy, we would use the **Set-SiteMailboxProvisioningPolicy** cmdlet. In this example, we change the max send size to 50.

```
[PS] C:\> Set-SiteMailboxProvisioningPolicy -Identity "SiteMailboxPolicy" -MaxReceiveSize 50MB
```

We can also mandate the prefix SharePoint uses when creating new mailboxes. By default, site mailboxes are prefixed with "*SM-*" in their name. If we wanted to change this prefix we could issue the **-AliasPrefix**

parameter. In this example, we change the default prefix to "*SiteMBX-*". Note that prefixes cannot exceed eight characters.

```
[PS] C:\> Set-SiteMailboxProvisioningPolicy -Identity "SiteMailboxPolicy" -AliasPrefix "SiteMBX-"
```

To view the characteristics of an existing policy you can use the **Get-SiteMailboxProvisioningPolicy** cmdlet.

If you want to locate all site mailboxes in Exchange, you can run the Get-Mailbox command with a recipient filter. In this example, we filter for all recipients with a type of team mailbox.

```
[PS] C:\> Get-Mailbox -RecipientTypeDetails Teammailbox
```

Note: For more information on site mailboxes check out this article https://technet.microsoft.com/en-us/library/mt577268(v=exchg.160).aspx

Arbitration Mailboxes

Arbitration mailboxes perform a variety of system tasks in the Exchange ecosystem. How many of these mailboxes you have will depend on which features you have enabled in your Exchange environment. However, every Exchange installation will have arbitration mailboxes regardless of feature set.

For example, email messages that are awaiting moderation are temporarily stored in the Microsoft Exchange Approval Assistant mailbox. Whereas the Microsoft Exchange Federation Mailbox is used when an Exchange organization is enabled with the Microsoft Federation Gateway. Other tasks performed by the arbitration mailboxes include offline address book generation, storing eDiscovery metadata and administrative audit logging.

These mailboxes are only visible in the Exchange Management Shell by running **Get-Mailbox** cmdlet with the **-Arbitration** parameter. To view the list of arbitration mailboxes in your environment issue the following command.

```
[PS] C:\> Get-Mailbox -Arbitration | Format-Table DisplayName, Name, Database -AutoSize

DisplayName                          Name                                                Database
-----------                          ----                                                --------
Microsoft Exchange Approval Assistant SystemMailbox{1f05a927-f637-41b6-b205-1257b1fa0d78} DB01
Microsoft Exchange                    SystemMailbox{bb558c35-97f1-4cb9-8ff7-d53741dc928c} DB01
Microsoft Exchange                    SystemMailbox{e0dc1c29-89c3-4034-b678-e6c29d823ed9} DB01
Microsoft Exchange Migration          Migration.8f3e7716-2011-43e4-96b1-aba62d229136      DB01
Microsoft Exchange Federation Mailbox FederatedEmail.4c1f4d8b-8179-4148-93bf-00a95fa1e042 DB01
Microsoft Exchange                    SystemMailbox{D0E409A0-AF9B-4720-92FE-AAC869B0D201} DB01
```

It is rare that you will manage arbitration mailboxes. The most common interaction with arbitration mailboxes is when you need to move them to a new database. This could be the result of a migration or when you are decommissioning a database.

To move all arbitration mailboxes from one database to another, issue the following command. The key here is the use of the **-Arbitration** switch which identifies all arbitration mailboxes in DB01. The results are then piped into a **New-MoveRequest** where we target the mailboxes to DB02.

```
[PS] C:\> Get-Mailbox -Database "DB01" -Arbitration | New-MoveRequest -TargetDatabase "DB02"
```

Discovery Mailboxes

Discovery mailboxes are used as a target for eDiscovery search results. Discovery mailboxes cannot send and receive mail. In addition, they cannot be converted to another mailbox type. This is intended to maintain the

integrity of the search data in the mailbox. Search results can be exported to a PST file for further analysis. Keep in mind that an administrator does not have permission to access the contents of this mailbox or perform an export by default. For these rights the individual performing the task would need to be assigned to the Discovery Management role group and the Import/Export role.

By default, Exchange ships with a default discovery mailbox. However, additional discovery mailboxes can be created if required.

To create a discovery mailbox, we run the **New-Mailbox** cmdlet. We designate this as a discovery mailbox with the **–Discovery** parameter.

```
[PA] C:\> New-Mailbox -Name "Rocket Design Legal Case" -Discovery
```

We then need to assign full access rights to the Discovery Management role group for this mailbox. By default, the Discovery Management role group only has full access to the default discovery mailbox. It is also possible to assign full access to other users and groups as well.

```
[PS] C:\> Add-MailboxPermission "Rocket Design Legal Case" -User "Discovery Management" -
AccessRights FullAccess -InheritanceType all
```

To locate all discovery mailboxes in the environment run the following command.

```
[PS] C:\> Get-Mailbox -Filter {RecipientTypeDetails -eq "DiscoveryMailbox"}

Name                          Alias                ServerName     ProhibitSendQuota
----                          -----                ----------     -----------------
DiscoverySearchMailbox...     DiscoverySearchMa...  EXCH01         50 GB (53,687,091,200 bytes)
Rocket Design Legal Case      G81d8a1579f304c2f...  EXCH01         50 GB (53,687,091,200 bytes)
```

To move all discovery mailboxes to a new database you can run a command such as this.

```
[PS] C:\> Get-Mailbox -Filter {RecipientTypeDetails -eq "DiscoveryMailbox"} | New-MoveRequest -
TargetDatabase "DB02"
```

Summary

In this chapter, we took a look at the various mailbox types. In addition, we discussed how to perform common administrative tasks through the Exchange Admin Center (EAC) and the Exchange Management Shell (EMS). Tasks that included the creation, deletion and management of mailboxes. We also looked at mailbox delegation and permissions, moving mailboxes and, the bulk creation and manipulation of mailboxes.

In the next chapter, we explore mail-enabled groups; including distribution groups, dynamic distribution groups and mail enabled security groups. We also explore mail-enabled users and mail contacts.