14 Migrations

In This Chapter

- Introduction
- Basics
- Checking Migration Status and Reports
- Public Folder Migrations

Introduction

It's very unlikely that there is no Exchange admin that has not or will not have to move one or more mailboxes from one Exchange database to another. While some scenarios are quite easy, there are some scenarios that require some more planning, reporting and so on.

With the introduction of Exchange 2010, Microsoft also improved the one element that would grow into an almost impossible task: Mailbox moves. The revolutionary change in Exchange 2010 made it possible to move mailbox data while the user still could access and modify his/her data: Online Mailbox Move. New incoming mail is queued in mail queues until the mailbox is available again (i.e. when it's successfully moved or has failed).

With the trend of growing average mailbox sizes, this was a necessary step. Otherwise it could mean that a migration would take too long to perform in a single big bang, meaning that you have to migrate mailboxes in stages and maintain a coexistence environment until the last mailbox has been moved. It was also a major step towards making Office 365 more accessible to migrate to and more flexible for Microsoft on managing servers and databases. Just consider moving mailboxes like in Exchange 2003, hoping that every mailbox has moved before your maintenance window closes… <shivers>.

Luckily this has changed, and as Exchange 2016 can only coexist with Exchange 2010 and 2013, earlier versions of Exchange won't be an issue. However, the option is still there with the -ForceOffline switch in the New-MoveRequest cmdlet. You shouldn't have to use it under normal conditions, however from time to time a mailbox is fickle and can only move via an Offline move.

Now, most of the move mailbox options are available from within the Exchange Admin Center in one way or another. But in our experience, EAC is probably fine for simple migrations or the incidental move of one mailbox. If you migrate your server environment from one major build to another, it's almost impossible to ignore PowerShell. Those migrations are far more complex and full of caveats, that it mostly always requires the use of custom PowerShell cmdlets and scripts.

But, before delving more into the PowerShell of (Online) Mailbox moves we shall explain the fundamentals of Mailbox moves since Exchange 2010.

Basics

Although Exchange 2016 creates Migration Batches (more on that later) automatically even when moving one single mailbox, the basis of a mailbox migration is the New-MoveRequest cmdlet. The name is telling; you request the system to move a mailbox. Why is that? Well, it could be the source or more importantly the target server is not in good health before or during the move. The Mailbox Replication service (present on each server in 2016) can decide that the move is too impactful or too risky and stalls the move.

Exchange is responsible for a successful mailbox move to be not to impactful on client experience, due to performance loss (a move does cost extra resources) and preventing data loss during a move.

Another benefit is that an Exchange server can reboot and the mailbox moves can resume after the server has rebooted, or another server will continue the move requests. This is possible since all move request are stored in arbitration mailboxes (system mailboxes, hidden from normal view) and another server could process them, if the servers are in a Database Availability Group (DAG). Or when a move fails for whatever reason, if you can resolve the issue you can restart the (online) move again. Or you could temporarily suspend any moves when you perceive any issues in your Exchange environment and after resolving those issues, continue at your leisure.

This means the mailbox migrations are a lot more robust and flexible for admins than in previous versions of Exchange (2007 and earlier). You can prepare, pre-stage the data and if necessary troubleshoot your migration long before completing the mailbox moves to the target servers. That moment that is traditionally prone to errors and requiring some aftercare normally. With online mailbox moves, these efforts will often be limited to client issues, rather than (also) server issues.

Migration Batch

As mentioned previously when using the web based Exchange Admin Center, even when you move one single mailbox it will create a Migration Batch. Migration Batches are bulk mailbox moves, which makes those bulk moves more easy to handle: Stopping or suspending them will stop/suspend all moves part of the batch.

Creating a most basic new batch:

```
New-MigrationBatch -Local -Name "Batch01" -CSVData ([System.IO.File]::ReadAllBytes("C:\scripting\batch01.csv")) -TargetDatabases "DB21"
```

The parameter Local indicates a move within the same Exchange environment (or AD Forest). Name is the identifier and CSVData is a CSV file with the mailboxes required to move with this Migration batch. TargetDatabases is the target database. The only column required in the CSV is "EmailAddress", where each mailbox to be migrated should have at least their email address listed, one per line. For information on the format of the input CSV file see: https://technet.microsoft.com/en-us/library/dn170437(v=exchg.150).aspx.

Although in this example there is only one database defined, it is possible to enter multiple target databases usable for this Migration Batch. You can do this by using the parameter formatting:

```
-TargetArchiveDatabases @("DB21","DB22","DB23")
```

However, you do not have control over which mailbox will end up in which database and it does not take mailbox sizes or current mailbox database sizes into account which mean you could end up with uneven distributed databases.

So, that's the basic command, but let's take a look at the other often used cmdlets and parameters. To know what's already present as a Migration Batch, you'd have to list them with Get-MigrationBatch:

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

Identity                       Status                 Type                   TotalCount
--------                       ------                 ----                   ----------
DB01 to DB02                   Completed              ExchangeLocalMove      2
Select Users                   Created                ExchangeLocalMove      3
```

With the AutoComplete parameter, you tell Exchange the mailboxes in the migration batch may be completed immediately when all the data has been moved of a specific mailbox. With the parameter AutoStart the batch will start at creation, if you do not use this you will have to start the batch manually with:

```
Start-MigrationBatch -Identity "<batchname>"
```

```
[PS] C:\>Start-MigrationBatch -Identity "Select Users"
[PS] C:\>
[PS] C:\>Get-MigrationBatch

Identity                       Status                 Type                   TotalCount
--------                       ------                 ----                   ----------
DB01 to DB02                   Completed              ExchangeLocalMove      2
Select Users                   Syncing                ExchangeLocalMove      3
```

You can see the status has (eventually) changed to Syncing, after starting the Migration Batch.

In optimal situations, there are no corrupt items in mailboxes, however based on years of experience there can be many unexpected corruptions that may make an object unreadable or unable to be migrated. Sometimes you can repair those objects, but it is not always practical. Luckily you can configure the Migration Batch to accept a number of corrupt items that will be skipped and thus lost, with the BadItemLimit parameter.

Not only can corrupt items stall a mailbox move, also large items that are over the set MaxSizeReceive value in the organization can halt a move. The LargeItemLimit specific the number of "violations" that are acceptable, but note that those items are not migrated and thus this can also lead to data/items being lost. You could consider increasing the MaxReceiveSize (and MaxSendSize), using Set-TransportConfig, in the organization or probably better these specific values on the mailbox in the source Exchange environment. Mailbox limits, like these, will stay in effect after the migration, because they are stored in the AD and replicated to Office 365 or the target forest normally. The BadItemLimit and LargeItemLimit parameters are available in the Exchange Admin Center as well, but if you require the use of Exchange PowerShell, these parameters are often a good practice to include. Note that the LargeItemLimit parameter is not valid for local moves, those within the same Exchange organization.

Example of the BadItemLimit parameter:

```
New-MigrationBatch -Local -Name "Batch01" -CSVData ([System.IO.File]::ReadAllBytes("C:\scripting\batch01.csv")) -TargetDatabases "DB21" -BadItemLimit 10
```

When your environment contains Archive mailboxes in Exchange Server, it might be required to move the primary mailbox separate from the Archive mailbox, you can configure that with the PrimaryOnly or ArchiveOnly

parameter. For instance, when you require only Archive mailboxes moved:

```
New-MigrationBatch -Local -Name "Archivebatch01" -CSVData ([System.IO.File]::ReadAllBytes("C:\scripting\Archivebatch01.csv")) -ArchiveOnly
```

You can use the latter one with the TargetArchiveDatabases parameter when you migrate only archive mailboxes. If you have multiple databases as a target, you can use the same syntax as with TargetDatabases:

```
-TargetArchiveDatabases @("DB21","DB22","DB23")
```

Whenever you don't use AutoComplete with your Migration batch, the batch will automatically synchronize each mailbox every 24 hours. If for whatever reason you do not want this to happen (you expect some performance issues during those synchronization moments for instance), you should use parameter AllowIncrementalSyncs with the value $False.

There are guidelines for the CSV files in order to work correctly with Migration batches. One requirement is a column with header EmailAddress that will contain the (primary) SMTP address of the mailbox you want to move. Other columns are optional, but when configured will overrule the settings made with MigrationBatch cmdlets. Optional column names are: TargetDatabase, TargetArchiveDatabase, BadItemLimit and/or MailboxType. The first three will act like previously described, MailboxType however will determine whether the primary, archive mailbox or both are moved (with the values PrimaryOnly, ArchiveOnly and PrimaryAndArchive).

A valid CSV would look like this:

```
EmailAddress, TargetDatabase, TargetArchiveDatabase, BadItemLimit, MailboxType
Manager@contoso.com, DB01, DB02, 10, PrimaryAndArchive
Admin1@contoso.com, DB04, DB04, 10, Primary
```

In this example: Manager primary mailbox will be moved to DB01, the Archive mailbox to DB02. Ten bad items are accepted. Admin1 primary mailbox will be moved to DB04, the Archive mailboxes will remain on it's current location. Ten bad items are accepted. See also this site for more info on CSV configuration for Migration Batches:

https://technet.microsoft.com/en-us/library/dn170437(v=exchg.160).aspx

If you get CSV files with some extra columns with attributes not used for mailbox migrations, you could use AllowUnknownColumnsInCsv in order to let Exchange ignore those and only focus on the Identity values.

For those who require regular updates, via email, on the status of the migration batch, use the NotificationEmails parameter. It's a multivalued string and entries should be delimited with commas. Like so:

```
New-MigrationBatch -Local -Name Batch01 -CSVData ([System.IO.File]::ReadAllBytes("C:\scripting\batch01.csv")) -NotificationEmails admin1@contoso.com, manager@contoso.com
```

Did you forgot a parameter or do you need to change a specific value? Know that you can change certain parameter values of existing Migration Batches with Set-MigrationBatch. For instance:

```
Set-MigrationBatch -Identity Batch01 -BadItemLimit 10 -AllowIncrementalSyncs $False
```

Not all values can be changed, for a full overview of the available Set-MigrationBatch parameters check out: https://technet.microsoft.com/en-us/library/jj218662(v=exchg.160).aspx.

For a full overview of the New-MigrationBatch parameters, what they do and when you should use them check out this page: https://technet.microsoft.com/en-us/library/jj219166(v=exchg.160).aspx

Other Migration Batch cmdlets are:

Start-MigrationBatch

- Starts a previously defined Migration Batch
- https://technet.microsoft.com/en-us/library/jj219165(v=exchg.160).aspx

Stop-MigrationBatch

- Stops a Migration Batch immediately, although already migrated (and potentially completed) mailboxes are left untouched.
- https://technet.microsoft.com/en-us/library/jj219168(v=exchg.160).aspx

Complete-MigrationBatch

- Finalizes the mailbox migration process. After this command has been entered, a last incremental synchronization will be performed and after that the user client will use the new mailbox location.
- https://technet.microsoft.com/en-us/library/jj218648(v=exchg.160).aspx

Remove-MigrationBatch

- Removes a non-running or completed Migration Batch. You can only have 100 Migration Batches at a time, so at some point it may be required to clean up old batches.
- https://technet.microsoft.com/en-us/library/jj219167(v=exchg.160).aspx

Get-MigrationConfig

- Too see the maximum number of batches or concurrent migrations.

Set-MigrationConfig

- Too change migration configuration settings, such as the maximum number of batches. You can change that from 100 to 200 with:
- Set-MigrationConfig -MaxNumberOfBatches 200