Another 10 useful PowerShell cmdlets for Exchange

By Drago on Friday, March 9, 2018

After all the positive feedback to my article "10 Useful PowerShell cmdlets for Exchange" (thanks again to Steve Goodman), I have decided to write a second part of it with another 10 useful PowerShell commands for Exchange and Office 365.

Same procedure as the last time. Some of you may already know the commands, but for some of you they are maybe new…

In any case, I wish you fun with reading this article…

This time, I will start with a command for Office365 and Exchange…

---

1 GET-MIGRATIONUSER

Migrating mailboxes from one Exchange to another, or even to Exchange Online is no rocket science anymore. How to do that, we can find in many articles in the Tech Blogs all over the world. I want to write here about how to check the status of the migration. All we need to do is to connect by PowerShell to the right endpoint. In this example, I am connecting to the Exchange Online by using the following command:

```
1 $UserCredential = Get-Credential
2 $Session = New-PSSession -ConfigurationName Microsoft.Exchange -ConnectionUri https://outlook.office365.com/powershell-livei
3 Import-PSSession $Session
```

After I am connected to my endpoint, I can check the status of the migration batch with the following command:

```
1 Get-MigrationUser -BatchId StagedBatch1 | Get-MigrationUserStatistics
```

As we can see in the screen shot, we get the status of any move request within our migration batch.

By knowing this command and understanding PowerShell, we are free to modify this command as well. Here´s another example:

```
1 Get-MigrationUser -BatchId BATCH | where {$_.Status -ne 'Completed'} | Get-MigrationUserStatistics
```

In this example, only those mailboxes will be shown, which don't have the status "completed". This is very helpful to keep an easy overview of huge migration batches .

2 GET-MESSAGETRACKINGLOG

By working as an Exchange administrator or in 2nd level support, one of the main tasks is to track messages. Microsoft has created the Exchange Toolbox for this.

If we start it, we can see a collection of tools, which help us to investigate some issues.

However, to check the Tracking Log there is a faster and easier way to do it.

We simply can use the following PowerShell command for that:

```
Get-MessageTrackingLog -Start "02/26/2018 08:23:00" -End "02/28/2018 17:00:00" -Recipients $recipientSMTP -Server $ExchServer
```

In my example, I was searching for all mails sent during a time range on a specific exchange server, which have been sent to a specific recipient:

However, there are also some other options for searching. Instead of searching by recipient, we are also able to search by:

- MessageSubject
- Reference
- Sender
- InternalMessageId
- MessageId

In the most cases we search for sender, recipient or message subject. You can see the output in the example above, with which we can work…

Another option is to create a Grid-View like I described in myother article about PowerShell commands.

Now, I want to show you a different option for creating nice reports:

If we use the following command:

```
Get-MessageTrackingLog -Start "02/26/2018 09:00:00" -End "02/27/2018 09:10:00" | ConvertTo-Html &gt; "C:\_DrPe\message-track.
```

We will get a report in form of an HTML file, which looks like the one below.

3 SET-CALENDERPROCESSING

The next cmdlet works on Exchange server 2016 an on Exchange online, we can use the cmdlet to modify calendar processing options for resource mailboxes, which include the calendar attendant, resource booking assistant, and the calendar configuration. With this cmdlet we are able to do many things, let's go for some examples…

There is a company which has, let's say, 10 meeting rooms. 9 of them are free to book for all employees, but one is only available for a dedicated group in the company, e.g the HR.

So now we need to prevent all users except the HR to be able to book this meeting room.

To do this, we simply run the following command:

```
Set-CalendarProcessing "MeetingRoom1" -BookInPolicy "User1","User2","User3","User4"
```

Now we are sure, that only these users are able to make bookings for the specified meeting room.

Another example automates the processing of calendar requests to the resource mailbox SBB23:

```
Set-CalendarProcessing -Identity "SBB23" -AutomateProcessing AutoAccept -DeleteComments $true -AddOrganizerToSubject $true -A
```

With the Set-CalendarProcessing cmdlet we are also able to disable the automatic processing. Here´s an example for a company car with the license plate GR123321:

```
Set-CalendarProcessing -Identity "GR123321" -AutomateProcessing None
```

4 SET-MAILBOXAUTOREPLYCONFIGURATION

Ready for number 4? This cmdlet, is again one for all the Exchange admins in the field. Sometimes it happens that users go to vacation and they forget to set the Out of Office Notification. Another example is, when a user is sick for a while and there is also a need to set the Out of Office notification.

There are different ways how to do that, but by some easy ways it could be a problem with the legal. To be fast to make it and to stay on the legal side, I prefer to user PowerShell for this task.

In my example the user Desmond Miles went to vacation and he forgot to activate his Out of Office notification. So, to be sure that people who are writing him will receive the right notification, I use the following command:

```
Set-MailboxAutoReplyConfiguration -Identity "Desmond Miles" -AutoReplyState Enabled `
-InternalMessage "I'm currently on leave until 23th April. Please contact Ezio Auditore on x72023 for urgent matters." `
-ExternalMessage "I'm currently on leave. Please contact our Administration Department on +41 12 345 67 89 for further assist
```

We can also schedule the time period from when till when we want to have the notification activated. Here, the way how this works:

```
Set-MailboxAutoReplyConfiguration -Identity "ALIAS" -AutoReplyState Scheduled –StartTime “02/28/2018 07:00:00” –EndTime 03/18
-InternalMessage "I'm currently on leave until 23th April. Please contact Ezio Auditore on x72023 for urgent matters." `
-ExternalMessage "I'm currently on leave. Please contact our Administration Department on +41 12 345 67 89 for further assist
```

---

5 NEW-MAILBOXEXPORTREQUEST

Here we go, halftime with the number 5…

The New-MailboxExportRequest cmdlet helps us to create a PST File from a user mailbox without the compulsion to login to the users account. This helps us, like the previous cmdlet, not to get in trouble with the legal. This command only works with on-premise Exchange servers, if you need to make a PST export from an Exchange Online mailbox, you can check my other article here.

Note: This cmdlet is available only for the Mailbox Import Export role and by default, that role isn’t assigned to a role group. To use this cmdlet, you need to add the Mailbox Import Export role to a role group (for example, to the Organization Management role group). Another requirement for exporting to a file share ist to grant the group ” Exchange Trusted Subsystem” read/write permissions to the target share.

In the first example we are going to create a PST export of the user mailbox “User01” to a file share server. To do that, we can run the following command:

```
New-MailboxExportRequest -Mailbox User01 -FilePath ‘\\SERVER01\PSTFileShare\User01_Recovered.pst’
```

How about an archive? The command is almost the same one! We only need to add the option “IsArchive”

```
New-MailboxExportRequest -Mailbox User01 -FilePath ‘\\SERVER01\PSTFileShare\User01Archive_Recovered.pst’ -IsArchive
```

Sometimes we don’t want to export the whole mailbox to a PST file. Imagine, we are just interested in messages with the words “company” and “phone”, which have been received before a specific date. For this, we can edit our command in the following way:

```
New-MailboxExportRequest -Mailbox User01 -ContentFilter {(body -like "*company*") -and (body -like "*phone*") -and (Received
```

This commands can help us with the preservation of evidence, for example…

At this point, I would like to point out the following: with great power comes great responsibility!!
This is a very powerful command. Be aware of it…

---

6 GET-PARTNERAPPLICATION

This command works for Exchange on-premise and for Exchange Online. With this command we are able to configure partner applications such as Microsoft SharePoint to access the resources of our Exchange environment.

If I want to get access to data in a system, I usually have to log in (authenticate) and I must have obtained the required permissions (authorization). For users, this is a well-known process that you need to log in to the Active Directory first and then log on to the SID and their group memberships, for example, to access file shares that have ACL’s associated permissions added to them. For the Exchange administration with RBAC, users are also used, but then they are granted permissions via roles.

Microsoft describes this function like:

When Exchange 2013 receives an access request from a partner application via Exchange Web Services (EWS), it parses the www-authenticateheader of the https request, which contains the access token signed by the calling server using its private key. The auth module validates the access token using the partner application configuration. It then grants access to resources based on the RBAC permissions

granted to the application. If the access token is on behalf of a user, the RBAC permissions granted to the user are checked.

So, here´s an example how this command works for us:

If we use the following command:

```
Set-PartnerApplication HRApp -RefreshAuthMetadata
```

We are refreshing the auth metadata for the HRApp partner application.

Another example is, when we want to create a new HRApp partner application called HRApp:

```
New-PartnerApplication –Name HRApp –ApplicationIdentifier 00000009-1234-12we-tz12-123654789wef -Realm contoso.com -UseAuthSer
```

The ApplicationIdentifier parameter specifies a unique application identifier for the partner application that uses an authorization server. When specifying a value for the ApplicationIdentifier parameter, you must also use the *UseAuthServer* parameter.

---

7 NEW-ITEM

Now let's talk about one we all know (I guess). Sometimes when I am writing a script I want to create a dedicated directory to, for example, dump out files or reports. When I write scripts alone or together with Dominic, I try to make them as universal as possible. Many things can be handled with variables, but sometimes we have to create things, in this case a directory.

To create a directory with PowerShell we simply need one command: New-Item.

With the New-Item cmdlet we can create different things like directories, files, registry keys, etc.

Here are some examples how to use it:

If we want to create a new directory on the C:\ partition for our PowerShell scripts, we can go with this command:

```
New-Item –ItemType Directory –Force –Path c:\DrPe\scripts
```

If we want to create a file in this directory, we can use almost the same command, for creating a .txt file we will use:

```
New-Item –ItemType File –Path c:\DrPe\scripts\demo.txt
```

How about existing files? If we try to create a new file but this file already exists, we will receive the following error message:

```
New-Item : The file 'C:\scripts\new_file.txt' already exists.
```

However, if we want to overwrite the existing File, we have the option *–Force* to overwrite the default behavior. Then the command will look like this:

```
New-Item –ItemType File –Path c:\DrPe\scripts\demo.txt –Force
```

---

8 NEW-MSOLUSER

Let's talk abit about the Office 365 cloud now. The next command is more known by using its bro*Get-MsolUser*.

I want to show you now, how to create new users in the Azure Active Directory. After we have made a successful login to the Azure AD throughPowerShell and successfully downloaded the right PowerShell module before, we can use the command New-MsolUser to create new users.

Now,I want to show you some examples, how we can do that…

If we need to create one new user, we can use the following command:

```
New-MsolUser -DisplayName &lt;DisplayName&gt; -FirstName &lt;FirstName&gt; -LastName &lt;LastName&gt; -UserPrincipalName &lt;
```

This is easy going, but to create a bulk of new user accounts there must be a easyier way… Yes, there is one!

First we need to create a CSV file. This can look like this:

```
UserPrincipalName,FirstName,LastName,DisplayName,UsageLocation,AccountSkuId
```

```
ClaudeL@contoso.onmicrosoft.com,Claude,Loiselle,Claude Loiselle,US,contoso:ENTERPRISEPACK
LynneB@contoso.onmicrosoft.com,Lynne,Baxter,Lynne Baxter,US,contoso:ENTERPRISEPACK
ShawnM@contoso.onmicrosoft.com,Shawn,Melendez,Shawn Melendez,US,contoso:ENTERPRISEPACK
```

After we have created this file, we can run the following command:

```
Import-Csv -Path &lt;Input CSV File Path and Name&gt; | foreach {New-MsolUser -DisplayName $_.DisplayName -FirstName $_.First
```

Now, PowerShell is doing magic things, and all users from our CSV file will be created.

With the New-MsolUser cmdlet we can also create new users and assign a license straight away:

```
New-MsolUser -UserPrincipalName "Desmond.Miles@contoso.com" -DisplayName " Desmond Miles " -FirstName "Desmond" -LastName "Mi
```

There are also a bulk of optional parameters:

```
-AlternativeEmailAddresses # Specifies alternative mail address for the user
-AlternativeMobilePhones # Specifies alternative mobile phone numbers fort he user
-BlockCredential # Specifies whether the user is not able to log on using their user ID
-City # Specifies the city of the user
-Country # Specifies the country of the user
-Department # Specifies the department of the user
-DisplayName # Specifies the display name
-Fax # Specifies the fax number of the user
-FirstName # Specifies the first name of the user
-ForceChangePassword # Indicates that the user is required to change their password at the next time they sign in
-ImmutableId # Specifies the immutable ID of the federated identity of the user
-LastName # Specifies the last name of the user
-LastPasswordChangeTimestamp # Specifies a time when the password was last changed
-LicenseAssignment # Specifies an array of licenses to assign the user
-LicenseOption # Specifies the options for license assignment. Used to selectively disable individual service plans within a
-MobilePhone # Specifies the mobile phone number of the user
-Office # Specifies the office of the user
-Password # Specifies  the new password for the user.
-PasswordNeverExpires # Specifies whether the user password expires periodically
-PhoneNumber # Specifies the phone number of the user
-PostalCode # Specifies the postal code of the user
-PreferredDataLocation # Specifies the preferred data location for the user
-PreferredLanguage # Specifies the preferred language of the user
-SoftDeletionTimestamp # Specifies a time for soft deletion
-State # Specifies the state or province where the user is located
-StreetAddress # Specifies the street address of the user
-StrongAuthenticationRequirements # Specifies an array of strong authentication requirements
-StrongPasswordRequired # Specifies whether to require a strong password for the user
-StsRefreshTokensValidFrom # Specifies a StsRefreshTokensValidFrom value.
-TenantId # Specifies the unique ID of the tenant on which to perform the operation
-Title # Specifies the title of the user
-UsageLocation # Specifies the location of the user where services are consumed
-UserPrincipalName # Specifies the user ID for this user
-UserType # Specifies the user type
```

---

9 GET-HOTFIX

So, the second to last for this article. The Get-Hotfix cmdlet gets hotfixes (also called updates) that have been installed on either the local computer (or on specified remote computers) by Windows Update, Microsoft Update, or Windows Server Update Services; the cmdlet also gets hotfixes or updates that have been installed manually by users.

In the first example we go to check which updates were already installed on my client computer. To do that I simply run the following command:

```
Get-Hotfix
```

And as we see in the print screen, there are just a few updates installed on my machine:

In the 2nd example I want to check which security hotfixes are installed on a list of computers. To do that we can use the following command:

```
Get-HotFix -Description "Security*" -ComputerName "Server01", "Server02" -Cred "Server01\admin01"
```

As we see, there are a couple of patches installed:

10 NEW-MANAGEMENTROLEASSIGNMENT

The (last) command for todays article is working for on-premise Exchange servers and for Exchange Online as well. With the New-ManagementRoleAssignment cmdlet we are able to manage roles for department groups. Maybe it sounds strange, but when I show you some examples, all will be clearer…

In the first example, I have to assign the Mail Recipient role to the 2nd level helpdesk of our company. To do that I run the following command:

```
New-ManagementRoleAssignment –Role ‘Mail Recipients’ –SecurityGroup ‘Level 2 Support’
```

We are also able to get some information about management roles. This example assigns the MyVoiceMail role to the “Sales end-users” role assignment policy. First, the IsEndUserRole property on the MyVoiceMail role is verified to be sure it’s set to $true, indicating it’s an end-user role:

```
Get-ManagementRole "MyVoiceMail" | Format-Table Name, IsEndUserRole
```

After the role has been verified to be an end-user role, the role is assigned to the “Sales end-users” role assignment policy.

```
New-ManagementRoleAssignment -Role "MyVoiceMail" -Policy "Sales end-users"
```

BONUS

While preparing this article, I came across another command, which I do not want to withhold from you. This command can be used in many ways, I want you to show a funny example for it.

Since PowerShell 3.0 we can use the Invoke-RestMethod. That means we are able to put curl in our PowerShell. Here, my example what we can do with it:

```
(curl http://wttr.in/Zurich -UserAgent "curl" ).Content
```

I am located in Zurich Switzerland, so for me, the weather here is important. When I run the command above, it will show me following result in my PowerShell console:

Go and play around with this command a little bit. An overview of available options can be found here: http://wttr.in/:help

I hope you enjoyed this article about another 10 PowerShell commands.

I hope you also have a sunny day like we here, as you can see in that last screen shot… But, hopefully just a bit warmer.

DRAGO

Drago is a Microsoft professional for Office 365, Microsoft Exchange, PowerShell and Cloud services. He works as senior System Engineer and Consultant in a leading swiss IT company and CSP. He is also a Trainer for Microsoft Cloud services and Web 2.0 in swiss schools.

Previous Post
5 Essential steps to keep all your mails safe

Next Post
Microsoft Cloud Services Expand in Europe including New Swiss Datacenters

Related Posts

EXCHANGE / OFFICE 365 / POWERSHELL

Powershell function for Exchange online

By Drago on Tuesday, February 12, 2019

EXCHANGE / OFFICE 365 / POWERSHELL

Allow Exchange users to see Calendar availability on G Suite

By Drago on Tuesday, March 19, 2019

ACTIVE DIRECTORY / POWERSHELL

.NET Assemblies In

PowerShell – Part 2: Manage Active Directory group members and user accounts

By Dominic on Thursday, March 14, 2019

ACTIVE DIRECTORY / EXCHANGE

Modify Mailbox type using Exchange Attribute Editor

By Drago on Tuesday, February 26, 2019

ACTIVE DIRECTORY / POWERSHELL

.NET Assemblies In PowerShell – Part 1: Manage Active Directory groups

By Dominic on

Tuesday,
February
19,
2019

EXCHANGE
/
OFFICE
365
/
POWERSHELL

Powershell function for Exchange online

By
Drago
on
Tuesday,
February
12,
2019

EXCHANGE
/
OFFICE
365
/
POWERSHELL

Allow Exchange users to see Calendar availability on G Suite

By
Drago
on
Tuesday,
March
19,
2019

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Submit

Yes, add me to your mailing list.

Search by Keyword

Search...

Navigate

- Checkout
- MyAccount
- Technet Links
- Guest articles
- Speaker Booking
- Meetup…
- Tech Events
- Photo Gallery
- dotCLOUD Messaging Suite 19

Subscribe to our Newsletter

Email *

Subscribe!

popular posts

Fixing Outlook Calendar Issue

Exchange PowerShell Suite – by MSB365

Import PST files using Exchange PowerShell

categories

Active Directory	(3)
ADFS	(11)
Azure	(18)
Azure Pack	(1)
CSP	(2)
DC/DNS	(3)
Events	(7)
Exchange	(86)
Intune	(2)
Monitoring	(1)
MS Teams	(2)
Office 365	(78)
OMS	(1)
PowerShell	(61)
Product Review	(2)
Social Media	(1)
VARIA	(16)

DRAGO PETROVIC

Drago is a Microsoft MCP for Office Servers and Services. He works as a consultant, Senior System Engineer and Analyst.

DOMINIC MANNING

Dominic Manning is a MCP for Microsoft Server and Services. He works as a Senior System Engineer with main focus on automation.

Be the first of your friends to like this

Twitter