10 | 19 Special print for ScriptRunner Software

IT Administrator
Das Magazin für professionelle System- und Netzwerkadministration

Managing Exchange with PowerShell

www.it-administrator.de

Quelle: seita – 123RF

Managing Exchange with PowerShell

Absence management

by Heiko Brenn

PowerShell is playing an increasingly important role in the management of Exchange servers. This is due to the fact that a number of functions are not even available via the Exchange Admin Center, and the fact that the use of the Admin GUI for regularly recurring tasks takes a lot of time. In this workshop we will look at which cmdlets can be used to implement absence management in Exchange.

PowerShell in Exchange 2019 allows you to perform many tasks easily, quickly, and in a standardized way. The topic of absence management is a good example of this, because setting up or adjusting the out-of-office settings for one or more employees is a constant challenge in the administration and helpdesk teams not only during holiday periods.

Although this workshop covers the on-premises variant of Exchange, the listed PowerShell cmdlets are also available for Exchange Online. This ensures that once created, one-liners and scripts can usually be used both locally and in hybrid and cloud environments in the same way.

Connect to the server

To use PowerShell to send commands to the Exchange server, you must first connect to it.

In our scenario, we assume that the Exchange administrator does not directly log on to the Exchange server via RDP, for example, but works on a Windows client. You also want to avoid installing additional PowerShell modules on the client. To achieve this, take advantage of the implicit remoting of PowerShell. You access the Exchange server directly and run the required cmdlets there.

First, use the New-PSSession cmdlet to connect to the Exchange.company.net server. To run these cmdlets successfully, you need a user account that has the necessary privileges on this Exchange server. Pass the credentials to the Get-Credentials cmdlet.

In the final step, you then use the Import-PSSession command to import the Exchange cmdlets into our local PowerShell environment. Thus, you can directly access the Exchange server from our current session.

Here is an example of the whole process:

```
$MyCred = Get-Credential

$Session = New-PSSession -Configura-
  tionName Microsoft.Exchange -Con-
  nectionUri http://Exchange.compa-
  ny.net/PowerShell/ -Authentication
  Kerberos -Credential $MyCred

Import-PSSession $Session -Disable-
  NameChecking
```

An initial test with the Get-Mailbox cmdlet shows the list of all Exchange mailboxes. Therefore, the connection to the server is successfully established and we can dedicate ourselves to the absence settings.

Query existing settings

The Get-MailboxAutoReplyConfiguration cmdlet is available to query the current settings. For example, you can display information about individual users, all users, or the users of a particular group or OU. To check the configuration of an individual user, use the "-Identity" parameter. To uniquely identify a user, you can use Name, Alias, Distinguished Name, or the e-mail address, for example:

```
Get-MailboxAutoReplyConfiguration
  -Identity thomas@company.net

Get-MailboxAutoReplyConfiguration
  -Identity Thomas
```

As a result you get the current status and see in our example that an absence (AutoReplyState: Scheduled) from "29/07/2019 (StartTime)" to "02/08/2019 (EndTime)" is configured for this user. You can also see that the absence message text is set for internal recipients (InternalMessage) and external recipients (ExternalMessage).

Now let's see how you can query absence settings for multiple users. To do this, use the PowerShell pipeline (|). With their help, individual commands can be easily

```
PS C:\test\WebService> Get-MailboxAutoReplyConfiguration -id Thomas@company.net

RunspaceId                       : 7482a3fb-bd42-4fd0-887a-d35cdc47a119
AutoDeclineFutureRequestsWhenOOF : False
AutoReplyState                   : Scheduled
CreateOOFEvent                   : False
DeclineAllEventsForScheduledOOF  : False
DeclineEventsForScheduledOOF     : False
EventsToDeleteIDs                :
EndTime                          : 02/08/2019 00:00:00
ExternalAudience                 : All
ExternalMessage                  : <html>
                                   <body>
                                   I'm out of the office from 7/29/2019 until 8/2/2019
                                   </body>
                                   </html>

InternalMessage                  : <html>
                                   <body>
                                   I'm out of the office from 7/29/2019 until 8/2/2019
                                   </body>
                                   </html>

DeclineMeetingMessage            :
OOFEventSubject                  :
StartTime                        : 29/07/2019 00:00:00
MailboxOwnerId                   : company.net/APAC/Singapore/Consulting/Thomas
Identity                         : company.net/APAC/Singapore/Consulting/Thomas
IsValid                          : True
ObjectState                      : Unchanged
```

Figure 1: This is what the out-of-office information queried with PowerShell looks like for an individual user.

linked together. This gives us a variety of options for executing PowerShell commands and scripts.

If the size of the Exchange environment is manageable, you can easily view the out-of-office settings of all users. To do this, call the following command:

```
Get-Mailbox | Get-MailboxAutoReply-
  Configuration
```

The result is a complete list of all absence configurations. In larger environments this list can be very long and confusing. If you are interested only in users who have currently scheduled an absence, add Where-Object to the above command. This limits the output to the mailboxes where the "AutoReplyState" is set to "Scheduled".

```
Get-Mailbox | Get-MailboxAutoReply-
  Configuration | Where-Object
  {$_.AutoReplyState -eq
  "scheduled"}
```

To see the active absences without an end time, add "-OR $_.AutoReplyState -eq "enabled" to "Where-Object". This configuration is usually found in the mailboxes of employees who have left the company. This enables you to send business partners a notification of the employee's departure with simultaneous information about the contact details of the successor. As a result, our command now looks like this:

```
Get-Mailbox | Get-MailboxAutoReply-
  Configuration | Where-Object
  {$_.AutoReplyState -eq "scheduled"
  -OR $_.AutoReplyState -eq
  "enabled"}
```

If you only want to know the names of the users who have activated an absence, adding "fl identity" helps:

```
Get-Mailbox | Get-MailboxAutoReply-
  Configuration | Where-Object {
  $_.AutoReplyState -eq "scheduled"
  -OR $_.AutoReplyState -eq "enab-
  led"} | fl identity
```

So that this information can be used for documentation purposes, for example, write the list with "Out-file -filepath C:\test\result.txt" in a text file:

```
Get-Mailbox | Get-MailboxAutoReply-
  Configuration | Where-Object {
  $_.AutoReplyState -eq "scheduled"
  -OR $_.AutoReplyState -eq "enab-
  led"} | fl identity | Out-file
  -filepath C:\test\result.txt
```

Performing Recurring Queries

Of course, our example is also suitable for a planned task. For example, you could create the list once a week and share it with a file share. Another option is to send an e-mail with this report to the personnel team on a regular basis. To do this, use the Send-MailMessage cmdlet. In the following example we send an e-mail with the file "result.txt" and the sender receives a delivery confirmation:

```
Send-MailMessage -From 'Admin <admi-
  nistrator@company.net>' -To 'Theo
  <theo@company.net>' -Subject 'Wee-
  kly report: Out-of-Office' -Body
  "Please find the list attached"
  -Attachments C:\test\result.txt
  -Priority High -DeliveryNotifica-
  tionOption OnSuccess, OnFailure
  -SmtpServer 'exchange.company.net'
```

To make the screen output of larger amounts of data easier to read, PowerShell also provides options. With "Out-GridView" you can create an interactive table of absence information and filter and sort it. If you take our initial example Get-Mailbox | Get-MailboxAutoReplyConfiguration and add "| Out-GridView" to it, you get a result as shown in Figure 2. Of course, you also have the possibility to specify which information the table should display. To do this, use Select-Object and specify the names of the desired fields:

```
Get-Mailbox | Get-MailboxAutoReply-
  Configuration | Select-Object
  Identity,Starttime,Endtime,Exter-
  nalMessage,InternalMessage |
  Out-GridView
```

Creating Absence Configurations

After you have already become familiar with some of the options for displaying existing absence settings, you are now concerned with the editing options for these settings. Let's take a closer look at the Set-MailboxAutoReplyConfiguration cmdlet.

Let's start by creating an absence notification for a user. Let's assume that this user has left the company and that permanent notifications are to be sent when e-mails are sent to this mailbox. The notification text should be identical for internal and external e-mails. The

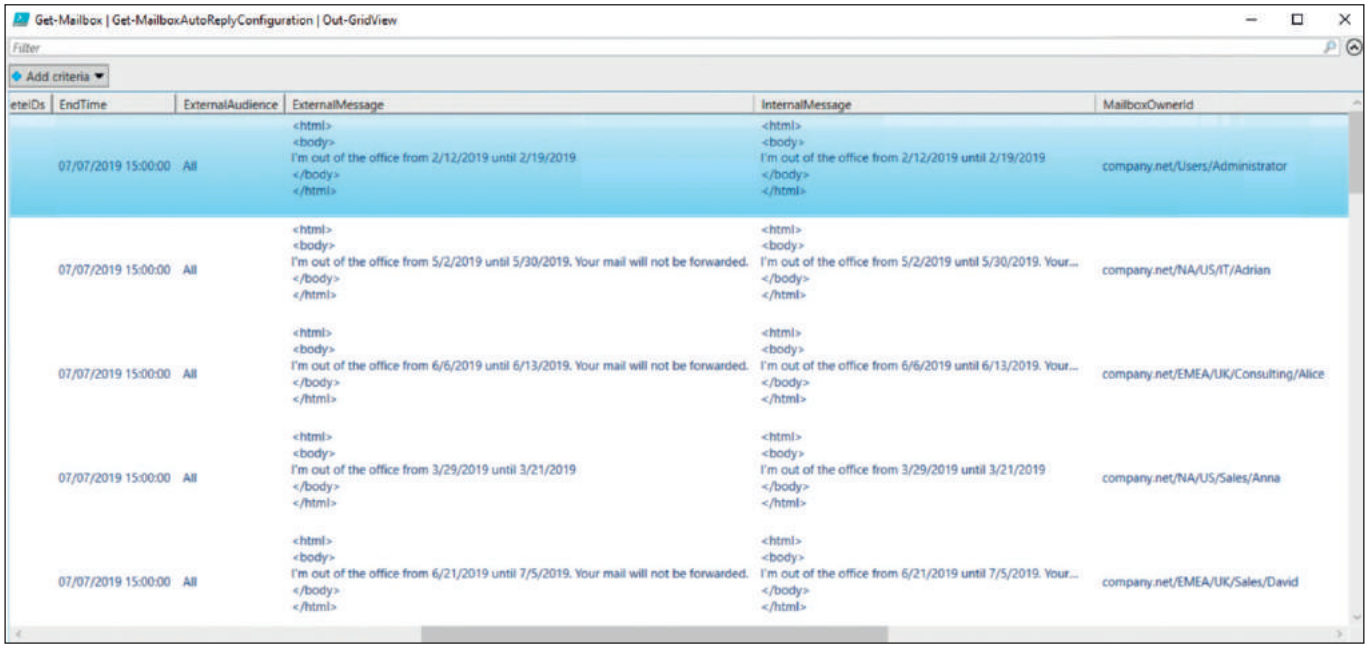

Figure 2: The list of absence settings as an interactive table.

corresponding command would then look like this:

```
Set-MailboxAutoReplyConfiguration
  -Identity Thomas@company.net
  -AutoReplyState Enabled
  -InternalMessage "This user is no
  longer working for us. Please
  write to Sandra@company.net"
  -ExternalMessage "This user is no
  longer working for us. Please
  write to Sandra@company.net"
```

In a second example, an employee has become ill and cannot take the absence himself, so the IT administration has to take over. Based on the sick report, an end time for the absence is now also known, and the out-of-office configuration can be configured accordingly:

```
Set-MailboxAutoReplyConfiguration -
  Identity Thomas@company.net -Auto-
  ReplyState Scheduled
-InternalMessage "I'm on sick leave
  until 12/07/2019. Your email will
  not be forwarded." -ExternalMessa-
  ge "I'm on sick leave until
  12/07/2019. Your email will not be
  forwarded." -EndTime "12/07/2019
  00:00:00"
```

If necessary, you can configure the messages for internal and external recipients differently.

For our third example, we want to create leave requests for multiple users. A scenario for this would be, for example, training for an entire team or a corporate event. For the implementation, we first need a file containing the users to be processed. To create it, use the following command, for example:

```
Get-Mailbox |Select-Object Prima-
  rySmtpAddress | ConvertTo-CSV
  -NoTypeInformation | Select-Object
  -Skip 1| %{$_ -replace '"',“”}|
  out-file C:\test\user.txt
```

We receive the primary e-mail addresses of all mailboxes as text files. With "Select-Object -Skip1" we eliminate the heading and with "%{$_ -replace """,""}" we remove the quotation marks.

You can now process the list using a ForEach loop. You create the defined absence configuration for each entry in the text file (i.e.: mailbox):

```
$Users = Get-Content
  C:\test\user.txt

$(foreach ($User in $Users)

{Set-MailboxAutoReplyConfiguration
  $User -AutoReplyState Scheduled
  -StartTime "08/09/2019" -EndTime
  "10/09/2019" -ExternalMessage "At
  the moment I'm attending a trai-
  ning event. I will answer your
  email asap." -InternalMessage "At
  the moment I'm attending a trai-
  ning event. I will answer your
  email asap."
})
```

Using HTML

So far, we have written the notification messages as pure text messages, but also the use of HTML is of course possible.

In this case, the HTML message text must first be saved in a file. The contents of this file are then passed to the cmdlet when you run Set-MailboxAutoReplyConfiguration.

In step 1, assign the contents of the prebuilt message to the $message variable:

```
$message = get-content
  C:\test\html_oof.txt
```

Then configure "$message" for external and internal messages:

```
Set-MailboxAutoReplyConfiguration
  -Identity Nora -AutoReplyState
  Scheduled -StartTime "08/09/2019"
  -EndTime "10/09/2019" -External-
  Message $message -InternalMessage
  $message
```

```
ExternalMessage    : <html>
                     <head>
                     <style type="text/css" style="display:none">
                     <!--
                     p
                      {margin-top:0;
                      margin-bottom:0}
                     -->
                     </style>
                     </head>
                     <body dir="ltr">
                     <div id="divtagdefaultwrapper" dir="ltr" style="font-size:12pt; color:#000000; font-family:Calibri,Helvetica,sans-serif">
                     <span style="color:rgb(255,0,0)">I'm on tour right now. <br>
                     </span>Your email will not be forwarded.
                     <div>Visit our website <a href="http://www.scriptrunner.com" class="OWAAutoLink">
                     www.scriptrunner.com</a></div>
                     <div><br>
                     </div>
                     </div>
                     </body>
                     </html>
```

Figure 3: HTML notifications can also be configured with the corresponding templates.

If you now run "Get-MailboxAutoReply-Configuration" for this user, you will see a result as shown in Figure 3. When the desired tasks have been completed, close the current session with Remove- PSSession -ID n. You get the ID of the session with Get-PSSession.

Identify possibilities for delegation

You have now learned how administrators can view and create absence settings. But what if these tasks are to be delegated to helpdesk staff or users in departments such as human resources? The following questions need to be answered first:

1. Can the employees handle the PowerShell?
2. May/can you assign these employees administrative authorizations for processing mailboxes?

If the answer to both questions is "Yes", you can easily delegate the above options in this form. If this is not the case, for example, because your colleagues do not have PowerShell know-how, or if the assignment of authorizations is not possible for organizational reasons, you must use special tools. These should have the following characteristics:

1. central execution and monitoring of all PowerShell scripts
2. central and secure storage of the required credentials to run the PowerShell
3. execution of PowerShell scripts also for non-PowerShell experts

These points are, for example, very easy to implement with the ScriptRunner software. All PowerShell scripts are centrally stored, managed, and executed.

In addition, this approach also allows PowerShell scripts to be developed together as a team. The required login information is securely stored in the Windows Credential Store or a password server. The user who is to create an absence configuration accesses the functions with a comfortable web interface and has no contact at all with the underlying PowerShell world.

The interface itself, by the way, is automatically created from the respective PowerShell script. This also completely eliminates the need for manual programming of GUI code. A further advantage results from the fact that variables for start and end of the absence are available in the absence notification.

These variables are automatically taken over from the date specifications. This also simplifies the creation of an out-of-office configuration and ensures that future notifications always leave the house with correct and up-to-date date information.

Conclusion

PowerShell is an excellent tool for easily performing repetitive Exchange administration tasks. With appropriate extensions, however, it is also a perfect basis for the secure and convenient delegation of defined use cases to employees and teams outside IT administration. (ln)

Heiko Brenn is Head of International Business at ScriptRunner Software GmbH.