Find-MailboxDelegates Guide

Table of Contents

[1. Setup 1](#_Toc485728905)

[2. Decide what parameters and switches to use 1](#_Toc485728906)

[3. Run the script – here are some examples 2](#_Toc485728907)

[Sample Input/Output Files 3](#_Toc485728908)

[Suggestion for extra-large environments 4](#_Toc485728909)

## 1. Setup

1. Download the powershell script from here: <https://github.com/alejandr0x0/Find-MailboxDelegates>
2. Copy the script to the Exchange server (or other domain joined machine) where you plan to execute the script. Note that **PowerShell Version 3 is required**.

## 2. Decide what parameters and switches to use

1. **What permissions to query?**

Extract only the permissions you require, this will help speed up the time to extract permissions. Note that Full Access permissions are now supported in cross premises, which means you don’t have to include those. We still include the capability however since AutoMapping is something that won’t work cross premises.

* 1. Full Access
  2. Send As
  3. Send On Behalf To
  4. Calendar Folder

1. **Which mailboxes to query against?**

You can run the script against ALL mailboxes in your environment or only a subset of mailboxes. In order for the batching of users to be most useful, you should run the script against ALL mailboxes so that all delegates are accounted for and not missed. Just run the script without the **-InputMailboxesCSV** parameter and it will query all mailboxes. Otherwise you can specify an input file with a list of mailboxes that you want to run the script against.

1. **Do you use security groups for permissions?**

Use the **-EnumerateGroups** switch so that the members of the groups (including nested) are included in the permissions output file.

1. **Do you have service accounts that have permissions to a lot of mailboxes?**

If you do, use the **-ExcludeServiceAccts** parameter so that these service accounts don’t end up linking all mailboxes together and you end up with one giant batch of users.

1. **Do you have security groups with permissions to mailboxes that you’re not interested in?**

If yes, use the **-ExcludeGroups** parameter so that the script doesn’t spend time enumerating these groups if they’re not needed.

1. **Want to execute the script against a particular exchange server?**

If yes, use the **-ExchServerFQDN** parameter to specify the FQDN of the exchange server you want to establish a session with.

1. **Need to resume a previous run that failed and not want to start from scratch?**

Use the **-Resume** switch and \*include\* the permissions you want to query so that the script will pick up on where you left off. The script will leverage the XML file in the same directory and find the mailboxes that are still in pending state and only run against those mailboxes and append the results to the existing permissions output file.

1. **Already have the permissions output file and only want to run the Batching logic (Step 2) and Create Migration Schedule (Step 3)?**

Use the **-BatchUsers** switch to run the script against the existing permissions output file in the same directory. The permissions output file must be the same name as the file that is output by the script otherwise it’ll fail.

1. **Already have the permissions output file and only want to run the Batching Logic (Step 2) BUT NOT Create Migration Schedule (Step 3) ?**

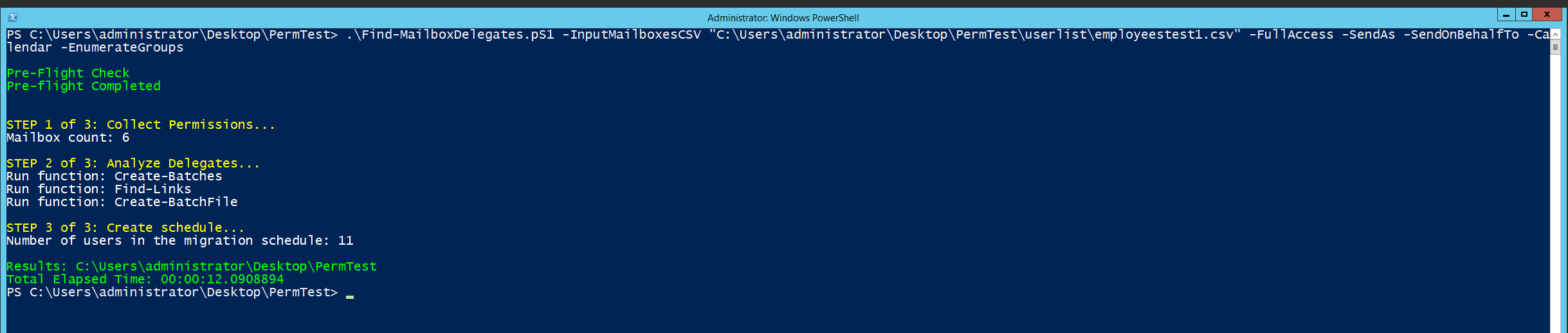
Use the -BatchUsersOnly switch to run the script against the existing permissions output file in the same directory. This will skip Steps 1 & 3.

1. **Only want to Get Permissions and skip the Batching Logic (step 2) & Create Migration Schedule (Step 3)?**

Use the -**GetPermissionsOnly** switch to only collect permissions. This will skip steps 2 & 3.

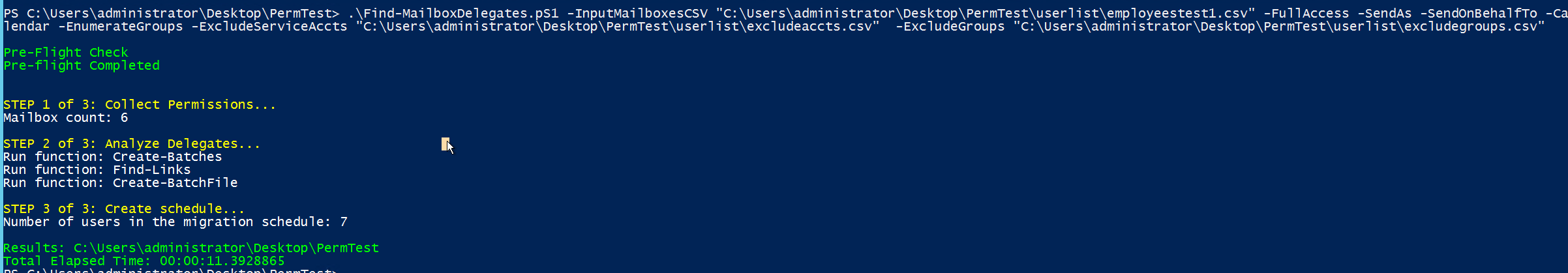
## 3. Run the script – here are some examples

**Example**: This will collect Full Access, Send As, Send On Behalf To, and Calendar Folder permissions for the list of users provided in the CSV file.



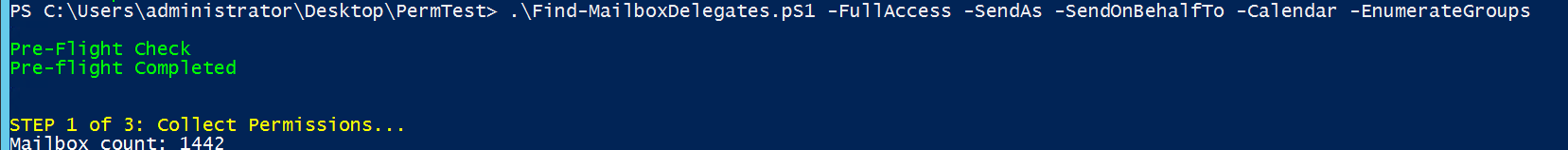
***.\Find-MailboxDelegates.ps1 -InputMailboxesCSV "C:\Users\administrator\Desktop\PermTest\userlist\employeestest1.csv" -FullAccess -SendAs -SendOnBehalfTo -Calendar -EnumerateGroups***

**Example**: This will collect Full Access, Send As, Send On Behalf To, and Calendar Folder permissions for the list of users provided in the CSV file and exclude service accounts ([jhay@mayalejo.com](mailto:jhay@mayalejo.com) and [qnikel@mayalejo.com](mailto:qnikel@mayalejo.com)) as well as security groups (testgroup and testgroupmailenabled).



***.\Find-MailboxDelegates.pS1 -InputMailboxesCSV "C:\Users\administrator\Desktop\PermTest\userlist\employeestest1.csv" -FullAccess -SendAs -SendOnBehalfTo -Calendar -EnumerateGroups -ExcludeServiceAccts “C:\Users\administrator\Desktop\PermTest\userlist\excludeaccts.csv" -ExcludeGroups "C:\Users\administrator\Desktop\PermTest\userlist\excludegroups.csv"***

**Example**: This will collect Full Access, Send As, Send on Behalf To, and Calendar Folder permissions for ALL mailboxes in the organization.



***.\Find-MailboxDelegates.pS1 -FullAccess -SendAs -SendOnBehalfTo -Calendar -EnumerateGroups***

## Sample Input/Output Files

**Input Files:**

|  |  |
| --- | --- |
| InputMailboxesCSV |  |
| ExcludeServiceAcctsCSV |  |
| ExcludeGroupsCSV |  |

**Output Files**:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Suggestion for extra-large environments

You can run multiple sessions of the script so that each session runs against a set of mailboxes and that way you can process more mailboxes in less amount of time. How many sessions you spin up will be up to you and the resource usage you’re comfortable with.

Here’s how you can do this:

1. Create multiple csv files with that have different lists of mailboxes. The number of csv files depends on the number of powershell sessions you will have going in parallel.
2. Spin up multiple powershell sessions and run the script pointed at the different CSV files using the **InputMailboxesCSV** parameter AND the **GetPermissionsOnly** parameter.
3. Merge the permissions output files (Find-MailboxDelegates-Permissions.csv) into one CSV file with the same name and put it in the directory of one of the powershell sessions you already have started. Any of the sessions will do.
4. In that PowerShell session, run the script with the following parameter **-BatchUsers** - this will bypass collecting permissions and jump straight into batching users using the merged permissions file you’ve provided as well as generate the Migration Schedule. You can also do -**BatchUsersOnly** if you prefer to only batch the users without creation the migration schedule. Make sure the permissions file is in the same directory as the script.