How to Use PowerShell to Automatically Assign Licenses to Your Office 365 Users

Windows Azure AD ScriptBox Item

The objective of this article is to introduce you to a process to assign licenses to your Office 365 users automatically.

This process is based on a set of PowerShell scripts.

You can find the required PowerShell script code at the end of this article.

The following table provides an overview of the related code files:

File	Description
SetupScript.ps1	Configures your environment for running the Licensing scripts. This script will prompt you for various parameters and will take use the three template files below to create customized scripts for your environment.
Get-LicensingInputFromAD.tmp	Creates a source data file based on information from your Active Directory Domain Service.
AssignLicense.tmp	Assigns licenses based on a source data file. The content of this template file is used by SetupScript.ps1 to create the actual PowerShell script file.
Get-MSOLUserLicensingReport.tmp	Creates a report of all licensed users in Office 365. The content of this template file is used by SetupScript.ps1 to create the actual PowerShell script file.

The description in this article assumes that these files exist in a subfolder of your choice on the hard drive of your computer.

Table of Contents

↑ Back to top

Setup of the scripted solution

In order to use this solution you should start by creating a subdirectory and place the four scripts below in a subdirectory of your choice. The SetupScript.ps1 will create the required subdirectory strucuture, prompt for O365 admin credentials and will generate the Powershell scripts AssignLicense.ps1, Get-LicensingInputFromAD.ps1 and Get-MSOLUserLicensingReport.ps1.

It will also prompt your for a DefaultUsageLocation, which is required in Office 365 for a license assignment. The script will use the country configured of a user in your AD by default. If this attribute is not configured - the DefaultUsageLocation you specify will be applied to the user. Please use the 2 letter country code from ISO-3166-1 alpha-2.

If you plan on using Active Directory as your source, you can use the included **SetupScript.ps1** script to configure the folders and scripts for use.

You can re-run the Setup script at any point to change the configuration.

If you want to run multiply copies of the script i.e. for group based assignment you can create multiple copies of the scripts in different folders and configure each script instance separately.

↑ Back to top

Understanding the Source Data Structure

To assign licenses, you need to provide the related source data that consists of the following components:

- An object identifier
- An operational attribute that describes the SKU you want to assign and the service plans you want to disable
- For group based licencense assignment, you need to provide an additional attribute on the group object to store the license information for the user account.
- For user based license assignement, you need to provide the following attributes:
  - UserPrincipalName
  - SearchFilter (e.g.: "@contoso.com")
  - an attribute that stores the licence information

In the operational attribute, the SKU you intend to assign is expressed in form of the SKU name.

Information: Currently this solution does not handle the assignment of product combinations like Office 365 and CRM Online, Project Online and Visio Online. The command set-msoluserlicense needs to be called per product (SKU) and the Assignlicense.tmp still needs to be modified between line 165 and 200 in order to handle this.

For example, the value of this attribute is "*DESKLESSPACK*" if you want to assign this SKU to a user.

The SKU you want to assign and the service plans you want to disable are separated by using the pipe symbol ("|").

For example, if a user should be assigned to the E3 suite but not to SharePoint Online, the value of your operational attribute is:

ENTERPRISEPACK|SHAREPOINTENTERPRISE|SHAREPOINTWAC

Note

Any non-licensing data in your employeeType attribute will result in no license set on that user, but it won't replace any existing license assignments.

Your source data is stored in a delimited text file that must have a specific header row, which is followed by the data rows.br> In this file, a semicolon is used as separator between the object identifier and the operational attribute.

The following shows an example for the content of a source data file:

```
userPrincipalName;UsageLocation;O365LicenseType
user1@contoso.com;GR;ENTERPRISEPACK
user2@contoso.com;DE;DESKLESSPACK
user3@contoso.com;US;ENTERPRISEPACK|MCOSTANDARD|SHAREPOINTWAC
```

↑ Back to top

Creating the Source Data File

The process outlined in this article requires you to create a source data file.

You can either create your source data file manually using the structure outlined in the previous section or you can use a script to retrieve data from your Active Directory Domain Service (ADDS).

Creating the Source Data File Using Your Active Directory Domain Service

If you want to use your on premise Active Directory Domain Service to create your source data file, you need to first populate the data about the SKU you want to assign and the service plans you want to disable.

As soon as you have populated the required data, you can create your source data file by using a script.

Preparing Your Active Directory Domain Service

In the section called "Understanding the Source Data Structure", you have been introduced to the required data format for the attribute that is used to store the information about the SKU you want to assign and the service plans you want to disable.

If you want to use your Active Directory Domain Service to store the related data, you need to populate an Active Directory attribute of the affected users with the required values.

The solution outlined in this article is based on a series of AD attributes that are used to locate affected objects and to store licence information.

Depending on your approach to assign license information, you might be required to set certain attributes in Active Directory prior to running the related script.

If you choose to create the licensing input file in some other manner, you can skip the following step:

Option 1: User based license assignment example

> Search Attribute ldapDisplayName i.e. userPrincipalName
> Search Filter i.e. *@contoso.com
> LicenseInformation Attribute ldapDisplayName i.e. extensionAttribute14

This retrieves all users in the specified domain that have the userprincipalName attribute set and where the attribute ends with "@contoso.com".

When the output file is created, the license information is added using the value from the extensionAttribute14 attribute.

The following shows an example for a related output file:

```
userPrincipalName;UsageLocation;LicenseType
user1@contoso.com;GR;ENTERPRISEPACK
user2@contoso.com;DE;DESKLESSPACK
user3@contoso.com;US;ENTERPRISEPACK|MCOSTANDARD|SHAREPOINTWAC
```

Option 2: Group based license assignment

> Search Attribute ldapDisplayName **memberOf**
> Search Filter i.e. CN=GroupName,CN=Users,DC=Contoso,DC=com
> LicenseInformation Attribute ldapDisplayName i.e. extensionAttribute14

This retrives all users in the specified domain that are DIRECT members of the specified group ().

When the output file is created, the license information is added using the value from the extensionAttribute14 attribute.

The folowing shows an example for a related output file:

```
userPrincipalName;UsageLocation;LicenseType
user1@contoso.com;GR;ENTERPRISEPACK
user2@contoso.com;DE;ENTERPRISEPACK
user3@contoso.com;UK;ENTERPRISEPACK
```

Creating the Source Data File Using a PowerShell Script

To create the source data file using a PowerShell script, you need to run the script called **Get-LicensingInputFromAD.ps1.**

You can find the code for this script at the end of this article in the section called "PowerShell Script Code".

The script stores the created data file in a folder called **queuedLicense**.

↑ Back to top

Assigning Licenses to Office 365 Users

The process of assigning licenses to your Office 365 users consists of two steps:

1. Connecting to your Microsoft Online Service Tenants
2. Assigning the licenses

Connecting to your Microsoft Online Service Tenant

Before you can run the **AssignLicense.ps1** script, you need to configure the scripts to allow them to connect to your Microsoft Online Service tenant.

To make the configuration you run the script called **SetupScript.ps1**.

This script does also create the required folder structure and it turns the two template script files (**AssignLicense.tmp**, **Get-MSOLUserLicensingReport.tmp**) into actual PowerShell scripts.

You can find the code for this script at the end of this article in the section called "PowerShell Script Code".

Note

You only need to run this script once per machine the scripts are intended to be run on. Run the SetupScript.ps1 script again if the password of the tenant admin account changes.

To connect you to your Microsoft Online Service tenant:

1. Logon to a Windows 7 or Server 2008 R2 Machine as an Administrator.
2. Create a folder called **O365LicenseScripts**.
3. Create all files from the "PowerShell Script Code" section in this folder.
4. Install the **Microsoft Online Sign In Assistant**.

5. Install the **Microsoft Online PowerShell Module**.
6. Run the **SetupScript.ps1** script:
    a. When prompted type the user name of a tenant administrator (i.e. admin@contoso.edu).
    b. When prompted type the password of the same tenant administrator.
7. Close PowerShell.

Note

If you have already created your source data file, you should copy it to the queuedLicense folder.

Assigning the licenses

To assign the licenses, you need to run the **AssignLicense.ps1** script.

The script moves the processed source data file into the **completedImportFiles** folder.

In addition to this, it creates a logfile in the logs folder and logs events in the Application EventLog.

Note

Not all users may be able to use all service plans in a SKU.

Please refer to our geography-specific restrictions here: http://www.microsoft.com/en-us/office365/licensing-restrictions.aspx#fbid=vztrPYJ44LA .

↑ Back to top

FAQ

Q: What if I would like to change the service plans within a SKU I already assigned to a user?

A: The licensing script does not support this scenario.

For example: Your user has the A3 Suite and you would like to assign the user the A3 Suite without Exchange Online.

We will not allow this reassignment.

You must manually remove the Exchange Online service plan from your user.

Q: What if I'd like to change between two SKUs?

A: The licensing script supports this scenario.

Q: What if I would like to assign more than one SKU at a time to the same user?

A: You will need to store your second SKU in another user attribute, and create a copy of these licensing scripts in a new folder.

Q: What if I try to license a user with a SKU I don't have? What if there's junk data in my AD attribute that I use for user license assignment?

A: This script will be unable to assign a nonexistent license to your user.

It will not remove any existing license on your user in an attempt to replace it with a nonexistent license.

↑ Back to top

PowerShell Script Code

The objective of this section is to provide the script code you need to complete the process outlined in this article.

SetupScript.ps1

```
# Script to Setup a new password encrypted string to put it into a script file
#Copyright Microsoft @ 2012
#DISCLAIMER
#The sample scripts are not supported under any Microsoft standard support progra
#The sample scripts are provided AS IS without warranty of any kind.
#Microsoft further disclaims all implied warranties including, without limitatior
#any implied warranties of merchantability or of fitness for a particular purpose
#The entire risk arising out of the use or performance of the sample scripts and
#In no event shall Microsoft, its authors, or anyone else involved in the creatic
#or delivery of the scripts be liable for any damages whatsoever (including, with
#damages for loss of business profits, business interruption, loss of business ir
#or other pecuniary loss) arising out of the use of or inability to use the sampl
#even if Microsoft has been advised of the possibility of such damages.
Clear
$AssignmentScriptTemplate = "AssignLicense.tmp"
$AssignmentScriptName="AssignLicense.ps1"
```

Get-LicensingInputFromAD.tmp

```
# Script to read the Licensing Information from Active Directory
# Copyright Microsoft @ 2012
# DISCLAIMER
# The sample scripts are not supported under any Microsoft standard support progra
# The sample scripts are provided AS IS without warranty of any kind.
# Microsoft further disclaims all implied warranties including, without limitatior
# any implied warranties of merchantability or of fitness for a particular purpose
# The entire risk arising out of the use or performance of the sample scripts and
# In no event shall Microsoft, its authors, or anyone else involved in the creatic
# or delivery of the scripts be liable for any damages whatsoever (including, with
# damages for loss of business profits, business interruption, loss of business ir
# or other pecuniary loss) arising out of the use of or inability to use the sampl
# even if Microsoft has been advised of the possibility of such damages.
#-------------------------------------------------------------------------------
# define the attribute containing the license information
# use the ldapDisplayname of the attribute, default is EmployeeType
```

AssignLicense.tmp

```
#script to assign or swap the licenses of a user
#Copyright Microsoft @ 2012
#DISCLAIMER
#The sample scripts are not supported under any Microsoft standard support progra
#The sample scripts are provided AS IS without warranty of any kind.
#Microsoft further disclaims all implied warranties including, without limitatior
#any implied warranties of merchantability or of fitness for a particular purpose
#The entire risk arising out of the use or performance of the sample scripts and
#In no event shall Microsoft, its authors, or anyone else involved in the creatic
#or delivery of the scripts be liable for any damages whatsoever (including, with
#damages for loss of business profits, business interruption, loss of business ir
#or other pecuniary loss) arising out of the use of or inability to use the sampl
#even if Microsoft has been advised of the possibility of such damages.
# Setup the UI Colors
$host.ui.RawUI.ForegroundColor = "White"
$host.ui.RawUI.BackgroundColor = "Black"
```

Get-MSOLUserLicensingReport.tmp

```
# Script to retrieve a licensing report from Office 365 and output it to CSV
# Copyright Microsoft @ 2012
# DISCLAIMER
# The sample scripts are not supported under any Microsoft standard support progr
# The sample scripts are provided AS IS without warranty of any kind.
# Microsoft further disclaims all implied warranties including, without limitatic
# any implied warranties of merchantability or of fitness for a particular purpos
# The entire risk arising out of the use or performance of the sample scripts anc
# In no event shall Microsoft, its authors, or anyone else involved in the creati
# or delivery of the scripts be liable for any damages whatsoever (including, wit
# damages for loss of business profits, business interruption, loss of business i
# or other pecuniary loss) arising out of the use of or inability to use the samp
# even if Microsoft has been advised of the possibility of such damages.
# change the username to a admin account in your tenant i.e. admin@contos.onmicrc
$Username="usernotset"
# to set a new password delete this script and run SetupScript.ps1 again.
```

Note

To provide feedback about this article, create a post on the AAD TechNet Forum.

For more FIM related Windows PowerShell scripts, see the AAD ScriptBox

AAD, AAD How To Script, Azure Active Directory, en-US, has code, has comment, has diagram, has FAQ, has image, Has Table, Has TOC, License assignment, magazine article, Office 365

Comments

Payman Biukaghazadeh 20 May 2013 8:35 PM

Revision: edited tags

Bryan Underwood 24 May 2013 7:14 AM

First of all, @Payman.... this script collection is pretty amazing. We are working from a large Seminary where student accounts area created every day automatically using a PERL script that talks to our Student Database program and creates new AD users, Exchange mailboxes, etc. Since I have certain AD containers setup to sync new users automatically, this script is a HUGE answer to what I was looking for to automatically assign licences to all of our Students. It looks very promising.

BUT... Everything seems to work fine up to the point of running the AssignLicenses.ps1 script.

I would like help with this error:

The string starting:

At C:\users\me\documents\O365LicenseScripts\AssignLicense.ps1:34 char:21

+ $script:Seperator = <<<< @"

is missing the terminator: "@.

At C:\users\me\documents\O365LicenseScripts\AssignLicense.ps1:306 char:1

+ <<<<

+ CategoryInfo : ParserError: ($("-" * 25)

... completed.`"

:String) [], ParseException

+ FullyQualifiedErrorId : TerminatorExpectedAtEndOfString

>> I am not a PowerShell whiz, so I tried following it's suggestion by adding [@.] at the end of the file, which is at line 306, Char:1 but this only caused the script to be accepted in PS but apparently not do anything at all. No log files, not errors, nothing.

>> The AssignLicenses.ps1 was created by running the SetupScript.ps1, which referenced an AssignLicences.tmp as a template.

>> BTW, the instructions above need to be updated to reflect the need to save the AssignLicences.tmp first, not AssignLicenses.ps1

Thanks,

Bryan

Bryan Underwood 24 May 2013 12:52 PM

Ok, first problem resolved:

Updated AssignLicense.tmp script: I had to remove the leading spaces on lines 36 and 41 where the ["@] script separator began. When you close a script separator it has to be in position Char 1 of the line.

New problem:

Running the AssignLicense.ps1 script, I am seeing this error dumped into the log file:

Cannot validate argument on parameter 'Message'. The argument is null or empty. Supply an argument that is not null or empty and then try the command again.. - Line:(134:26) $users|foreach-object {

5/24/2013 3:00:15 PM: License Update completed.

I cannot figure out what this error means! Again, I am not a PowerShell expert so this is a bit discouraging for me. It appears to be a syntax error but I am just not sure. Any help you could provide would be most appreciated.

mderosia 3 Jun 2013 1:39 PM

Thank you very much for sharing this script. I wish this information was alot easier and built into the portal because we also need to assign different licenses to staff and students.

Bryan, were you able to figure out the parameter issue inn the script because i see the same thing. I am not sure what needs to be done to correct this issue and it seems to be the same error that your getting.

My error is:

A positional parameter cannot be found that accepts argument '+'.. - Line:(134:27) $users|foreach-object {

Thanks,

Mark

Jesper Stahle 12 Jun 2013 6:55 AM

This is great. However, I really think that much of this functionality should be built into the Office 365 Admin portal as well. The filter options that you can use to list users and bulk assign licenses in the portal today could be far more extended and advanced.

Jethro SEGHERS 22 Nov 2013 12:09 AM

I love the scripts but there are some major caveats about it.

1. Usage Location needs to be set before you can Add a license

2. For some reason, $er in the AssignLicenses make alot of cmdlets fail

3. The scripts should be easy downloadable

4. @ and whitespaces.

5. Extend script for multiple licenses, it's not only office 365 licenses anymore. Project Online, CRM Online, ...

I'm rebuilding them for myself now, if you want I can send them to you ..

Philo Janus 13 Jan 2014 3:03 AM

Just one minor note - "automatically" means "without user intervention." You really can't say "automatically provision with a script" because then it's not automatic. Microsoft needs to provide a way to simply configure in Office 365 "provision any user in [AD Group] with [O365 License]" - *that* will be automatic.

It's really sad that MSFT is pitching this as an enterprise solution when there are so many "mom & pop" limitations.

DarrenBonehill 5 Feb 2014 6:16 AM

I have been trying to implement this and at the moment continually failing.

It looks as though the AssignLicense finishes but the license isn't getting set in the tenancy.

The Get-MSOLUserLicensingReport has errors but seems to come back with data from the tenancy and the licensing information.

Jethro - I hope you don't mind me asking but did you get your scripts working especially around multiple SKU's as this would be something I would be interested in.

Darren

The Real Don Balmer 17 Mar 2014 6:38 AM

I was having the same issue as others and can add that you should remove the preceding $ from all of the

-ErrorVariable $er switches to read -ErrorVariable er.

I'n no PS expert but believe this to be an error as you set a variable by name BUT refer to it using the variable prefix $

I also added the following comand at line 214 BEFORE I assign the license (I believe you need to set the location before you can assign the license).

'Set-MsolUser -UserPrincipalName $UPN -UsageLocation "GB" -ErrorVariable er '

The next line should be the license application code as per the code supplied.

Set-MsolUserLicense -UserPrincipalName $UPN -AddLicenses $NewLicenseExc -LicenseOptions $licenseOptionObject -ErrorVariable er

In case you skipped over/missed the other comments you need to remove the white spaces from the start of lines 36 & 41 and the file name should be AssignLicense.tmp

Thanks for the code by the way, it was a big help.

Now what would be REALLY nice is if it could differentiate between disabled plan options. For example I am deploying our licenses with Lync disabled (for now), your script can handle this of course BUT during evaluation of the current license you don't evaluate disabled options. A match on the applied license is not enough and skips, even though the license is 'technically' not the same as at some point I will add the Lync option

timbos - MSFT 15 May 2014 1:54 PM

Since I urgently needed a solution like this I have updated the scripts and the description. As mentioned on the comments below there were a few minor glitches in the script.

Furthermore the UsageLocation was not handled by the solution- I have added this as well.

Please apologize the weird formatting of the code - I focused on the script not on getting the HTML right on this WIKI. :)

In order to "automate" this solution you need to create scheduled tasks for both the GetLicensingInputFromAD.ps1 and the AssigenLicense.ps1 script.