AzureAD - Exchange to OfficeSpace Import

Introduction

This document describes the process of importing AzureAD users into your OfficeSpace instance. To help accomplish this, you can use the OfficeSpace-provided scripts.
The general use case would be to have the script run periodically, as a scheduled task, using an AzureAD service account.
The AzureAD - Exchange import script has support for sourcing employee photos from AzureAD, Exchange Online/Office365, or both.
Feel free to use and modify the script as you see fit.
Tip: It could be very helpful if you can first get a basic PowerShell script to run as a scheduled task, by the user you want it to run as. Once you have got that all working, create the OfficeSpace import task.

Environment

The process of getting users imported from AzureAD requires a compatible environment to run the scripts from. The following table describes these.

Table: Script Requirements

Requirement	**Comments**
A Windows system that is joined to your AzureAD domain.	I used a Windows 10 Pro 64-bit host.
PowerShell	My test systems had versions 5.1.17134.407, 5.1.17763.503 of the PSDesktop edition.
The “AzureAD” PowerShell module	I needed to install this, and used the current versions available at the time (2.0.2.4, 2.0.2.16). To install it, I ran powershell.exe as an Administrator, then: `Install-Module AzureAD` As of November 2018, this module was only available for the standard PowerShell edition (not PowerShell Core).
Ensure that scripts are allowed to run (set Execution Policy as needed)	See: https:/go.microsoft.com/fwlink/?LinkID=135170

	I ran a PowerShell session as an Administrator, then: `Set-ExecutionPolicy RemoteSigned`
Network access	Outbound access to AzureAD (to gather the user data) and outbound https to your OfficeSpace Software instance (to import the users).

Scripts

There are 2 scripts involved:

1) **push_from_o365_to_officespace.ps1** - This is the main PowerShell script that performs the import of users from AzureAD to your OfficeSpace instance. If you want to include employee photos, the photos can be sourced from AzureAD, Exchange/Office365, or both. The script can be added as a scheduled task.
2) **pwencrypt.ps1** - This script generates a password file. It should only need to be run once, and should be run by the user that will be running the import script.

Configuration for push_from_o365_to_officespace.ps1

Open the script in an editor. The configurations that you will need to edit are located at the top of the script.
See the following table for variables you will need to configure.

Table: Script Variables

Variable	**Comments**
$promptForCreds	If $true, the script will popup a window prompting you to enter your username and password. If $false, it will rely on $azureUsername and $credsFile. If the **-creds** command line option is provided, $promptForCreds is ignored.
$azureUsername	An Azure AD account that can access your directory. It should be in the format:

	username@yourdomain.com. This account will also be used to authenticate with Exchange Online/Office365 if you want to source photos from there. Note: If you are testing manually, and $promptForCreds = $false, then this variable is not used.
$credsFile	The path to your encrypted credentials for the azureUsername. You'll create this file when you run the "pwencrypt.ps1" script. Note: If you are testing manually, and $promptForCreds = $false, then this variable is not used.
$useLogFile	Controls whether the output will be logged to file ($logFile) or not.
$logFile	The path to a log file. The script will log output to this file, and will overwrite it each time it runs. $useLogFile should be set to $true to use this.
$photoSource	Where to look for employee photos. Supported values: **azuread** - Query AzureAD for user photos. **exchange** - Query Exchange Online/Office 365 for user photos. **exchange-azuread** - Query Exchange Online/Office 365 for user photos, and if no photo is found for the user, then query Azure AD. **none** - Don't attempt to use any photos.
$photosDir	The path to a location where the script can download AzureAD user thumbnail photo images to. If the directory doesn't exist, the script will attempt to create it, or you can simply create it ahead of time.
$ossToken	Your OfficeSpace instance API key
$ossHostname	The FQDN of your OfficeSpace instance

$tryNicknames	If $true and the user doesn't have a givenName configured, try to assign a nickname based on their displayName.
$importThreshold	Minimum percentage of import count compared to existing OSS record count. If the user count to import is less than this threshold %, don't import. This is intended to handle instances where the source data set retrieved is very small compared to the OSS data set.

Screenshot: Example configuration

```
##############
# User config
##############
$promptForCreds  = $false                                   # $true: will prompt user for credentials
                                                            # $false: will use creds stored in $credsFile
                                                            # If -creds command line option is provided, $promptForCreds is ignored.
$azureUsername   = "oss_service@coolco.com"                 # user with Azure AD access
$credsFile       = "C:\ps\oss_service.creds"                # encrypted user credentials file
$useLogFile      = $true                                    # Log output to file ($true=log, $false=do not log)
$logFile         = "$PSScriptRoot\oss_import.txt"           # Path to log file
$photoSource     = "exchange-azuread"                       # source for photos: azuread, exchange, exchange-azuread, none
$photosDir       = "C:\ps\photos"                           # directory to download thumbnail photos
                                                            #    (only used if $photoSource = 'azuread' or 'exchange-azuread')
$ossToken        = "84840f40f11222333de163a7f08f0fa1"       # OfficeSpace API key
$ossHostname     = "coolco.officespacesoftware.com"     # OfficeSpace instance hostname
$tryNicknames    = $false                                   # look at displayName for possible preferred name
$importThreshold = 60                                       # minimum percentage of import count compared to existing OSS record count
                                                            #   If the user count to import is less than this threshold %, don't import.
##############
```

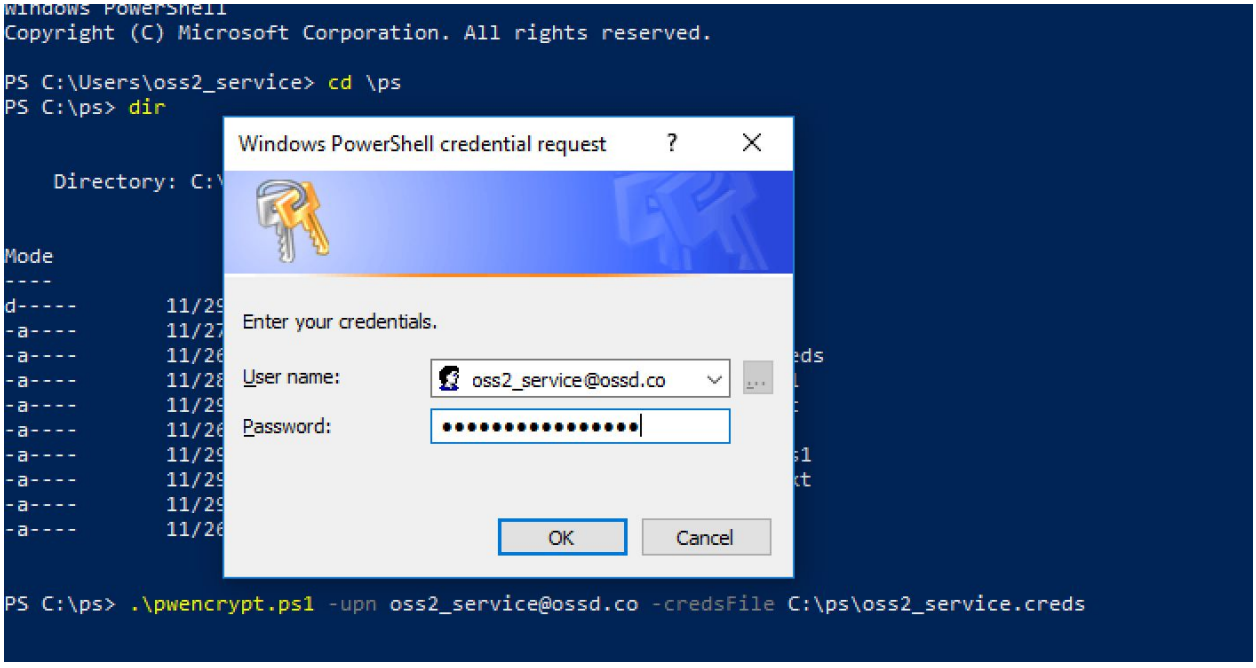
Step by Step Instructions

- Identify which Windows host that will run the OSS script.
- Ensure that your environment meets the script requirements (see table above).
- Identify which AzureAD user the script will run as. It can run as a regular user (service) account. As it is only reading AzureAD information, it doesn't need any special role assigned. However, if you plan to use a regular user (service) account to access Exchange Online/Office 365, then you will need to assign a role to the account.
    - To add permission for a service account to access mailboxes other than it's own, go to the Exchange admin center
    - Select permissions > admin roles and select "View-Only Organization Management"
    - Click on Edit (pen) and scroll down to Members.
    - Click on "+" to add a member. Select the user you want and click on the "add ->" button. [OK] [Save]
- Make sure the host is joined to your AzureAD environment.
- If you will be running the script as a regular user, add the user's account to the Windows host:

    - As an Administrator user:
      Settings > Accounts > Other people
      Add a work or school user
      User account: username@yourdomain.com
      Account type: Standard User
      click [Add]

  This then puts their AzureAD\username under the list of “Other People”
- Make sure you can run PowerShell scripts on this system (see the Requirements section).
- Sign in to your Windows host that will run the OSS script from. Use the username that will run the scripts.
- Copy the “push_from_o365_to_officespace.ps1” and “pwencrypt.ps1” scripts to the host.
- From a PowerShell session, run: `.\pwencrypt.ps1 -upn` *`username@yourdomain.com`* `-credsFile` *`c:\path\to\create\your_creds_file`* (replacing the -upn and -credsFile values to your own).
- This should open a window prompting you for your credentials (based on the UPN you provided). Enter the user’s password in this window.
- NOTE: The script does not check that the password is correct or not. It basically writes out the password to an encrypted standard string. The OSS user import script will use this credential file for authenticating to AzureAD without being prompted for a password.

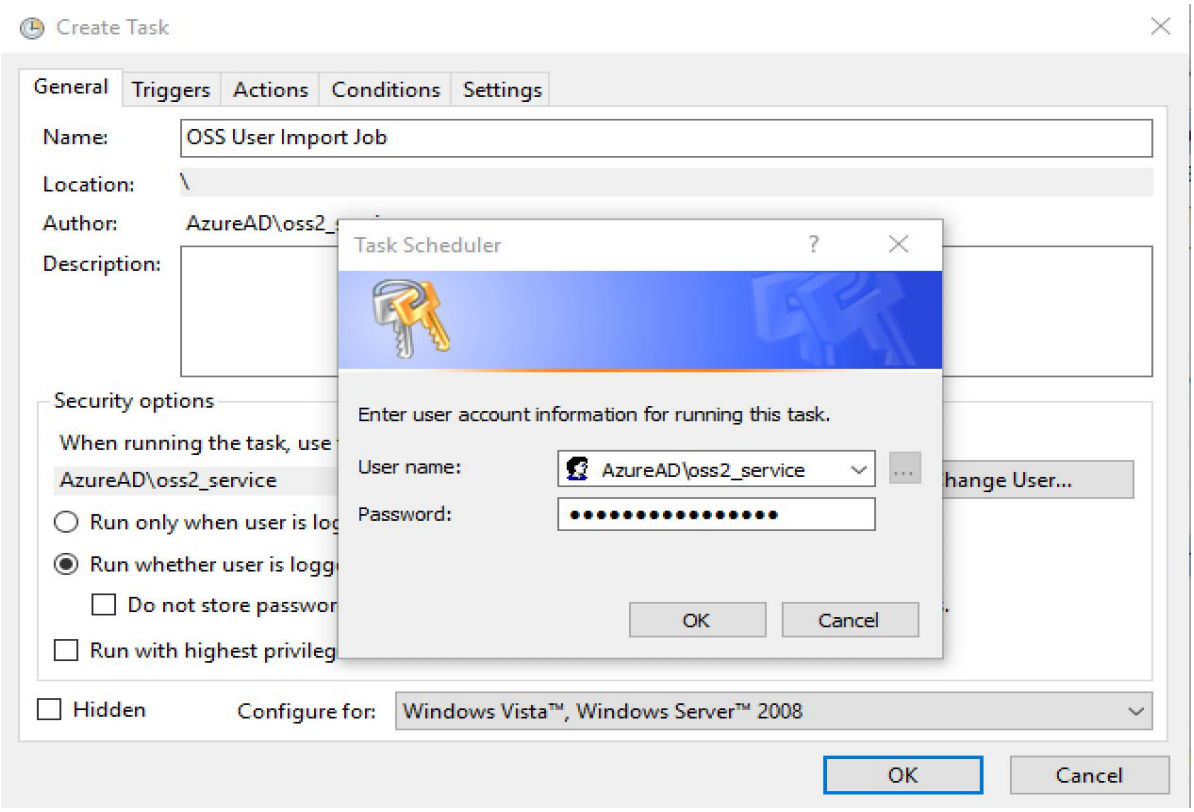
Example: Running the pwencrypt.ps1 script



- Edit “push_from_o365_to_officespace.ps1” configuration as needed (see the “Configuration for push_from_o365_to_officespace.ps1” section).

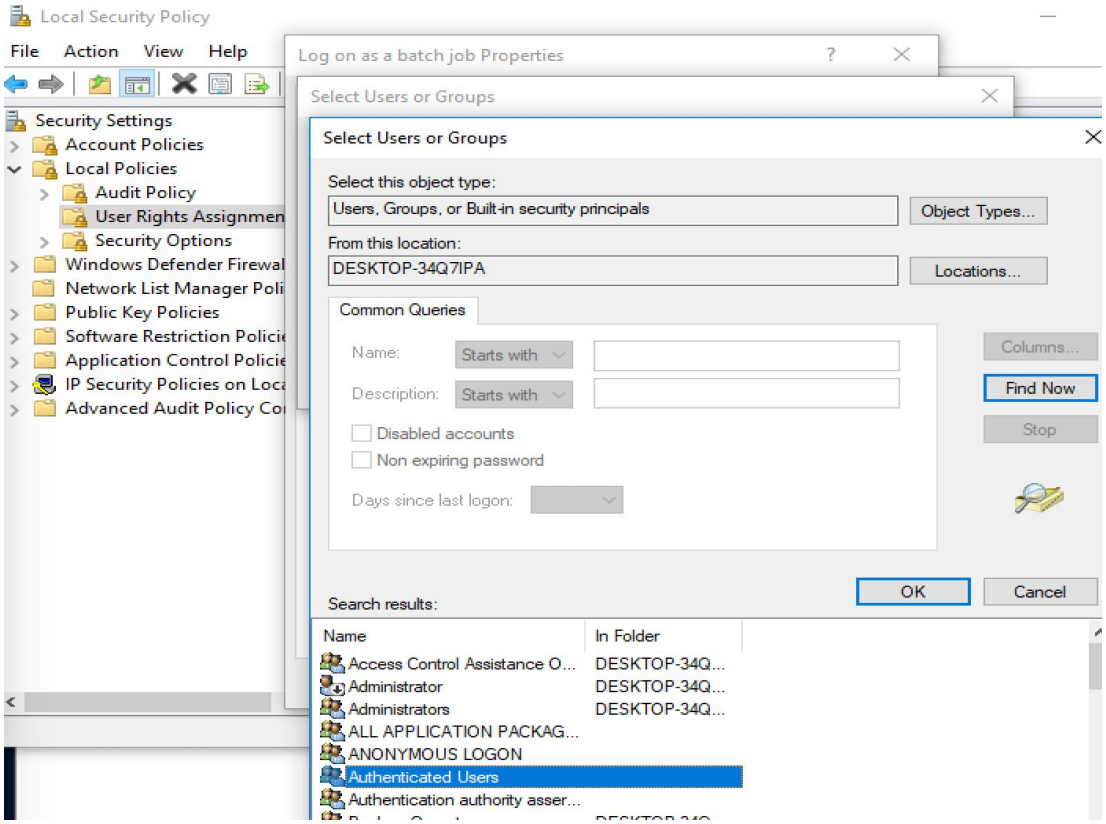
- When you want to add the script as a scheduled task, open Task Scheduler and create your task.
  Some items to ensure:
  (o) Run whether user is logged in or not
  Action: Start a program
  Program/script: powershell.exe
  Add arguments (optional): -f C:\path\to\push_from_o365_to_officespace.ps1
- When you click [OK] to save the task, you will probably get a popup window for the login credential for your service account.
  The username should be in the format: AzureAD\username
- Enter the password for the user.

Screenshot: Prompted for credentials when saving the scheduled task

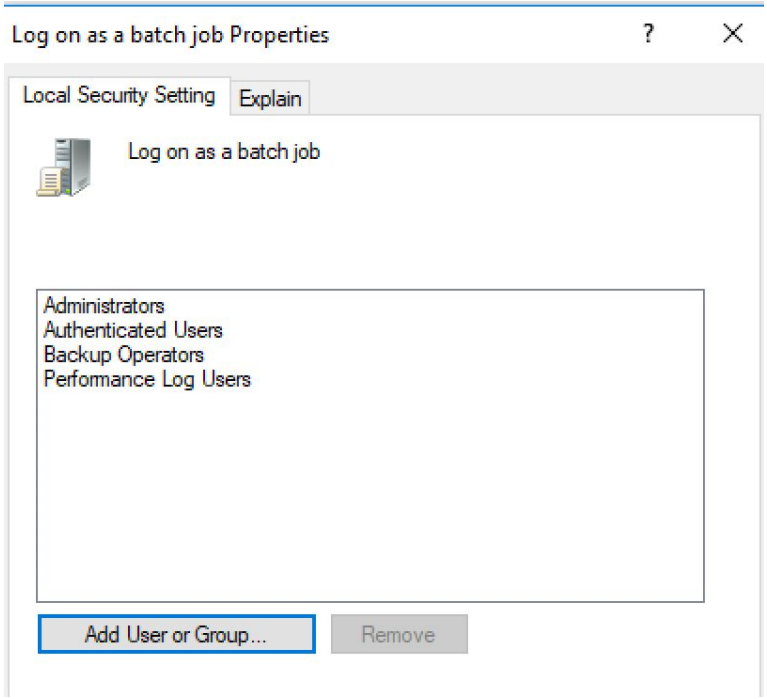


- If you get the message:
  This task requires that the user account specified has Log on as batch job rights. For more information about setting this policy, see the Task Security Context topic in Help.
  You can assign “Authenticated Users” the right to “Log on as a batch job”.
  As an Administrator, open the “Local Security Policy” application, and navigate to:
  Local Policies > User Rights Assignments > Log on as a batch job.
  See the below screenshots for more information.

Screenshot: Local Security Policy - adding Authenticated Users



Screenshot: Log on as a batch job Properties (after adding Authenticated Users)



Phew! That's it. Happy importing!
Table: AzureAD user filtering

Filter	**Description**
Get-AzureADUser -All $true	Return all users
Get-AzureADUser -All $true -Filter "AccountEnabled eq true"	Return all users that are enabled (skip disabled accounts).
Get-AzureADGroup -all $true | ? displayName -eq "FTE" | Get-AzureADGroupMember -All $true	Return all users that are members of the "FTE" group.
Get-AzureADGroup -all $true -Filter "DisplayName eq 'Finance'" | Get-AzureADGroupMember -All $true	Return all users that are members of the "Finance" group. This is another method to get members of a group.
Get-AzureADGroup -all $true -Filter "DisplayName eq 'FTE' or DisplayName eq 'Finance'" | Get-AzureADgroupmember -All $true	Return all users that are members of the "FTE" or "Finance" groups.