

# Powershell – podstawy automatyzacji w Office 365 dla EDU

Przygotował  
Konrad Sagała

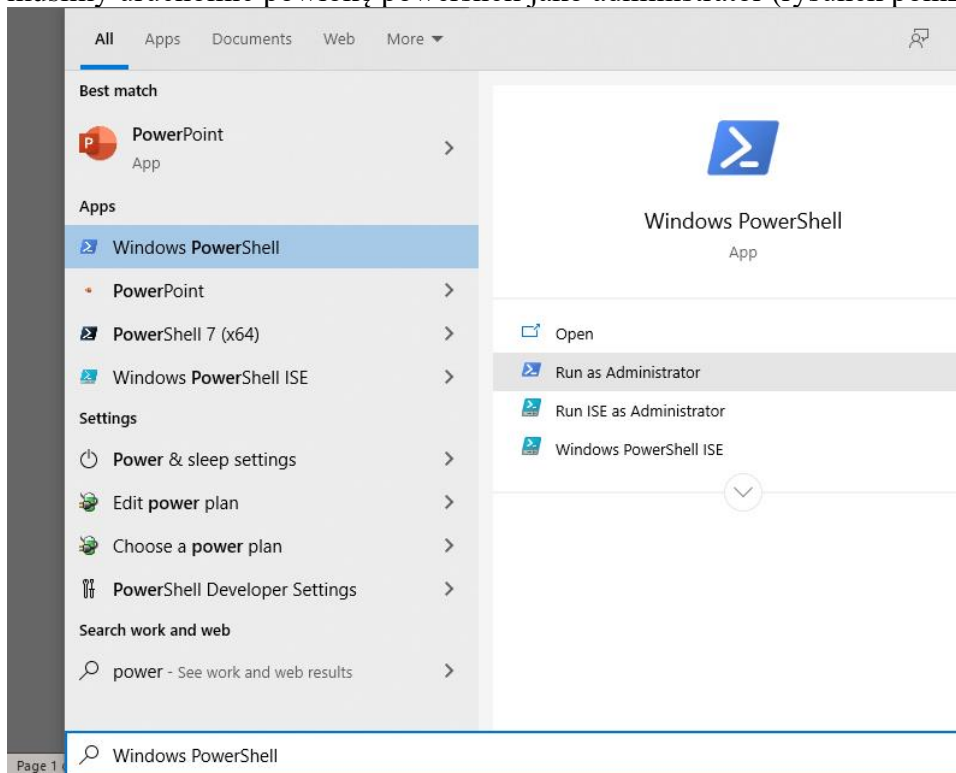


Chciałbym poruszyć temat automatyzacji niektórych zadań przy pomocy Powershella. Temat przygotowania swojego komputera do zarządzania środowiskiem Office 365 poruszałem na swoim blogu już kilkakrotnie, więc żeby się nie powtarzać zachęcam do lektury tych artykułów:

[Zarządzanie chmurą z powershell](#)

[Zarządzanie chmurą z powershell - ładowanie modułów](#)

Oczywiście moduły się zmieniają, pojawiają się nowe (np. dla Exchange Online czy Microsoft Graph), inne wychodzą z użycia. Warto więc co jakiś czas sprawdzać nowości w tym zakresie czy chociażby co jakiś czas uruchomić komendę update-module. Warto również pamiętać o tym, że domyślnie środowisko Powershell ma domyślnie zablokowaną możliwość wykonywania skryptów, które nie są podpisane cyfrowo, więc jeżeli chcemy uruchomić opisane dalej skrypty, musimy uruchomić powłokę powershell jako administrator (rysunek poniżej).



Następnie musimy zmienić politykę wykonywania skryptów wykonując komendę:  
`Set-ExecutionPolicy -ExecutionPolicy RemoteSigned`

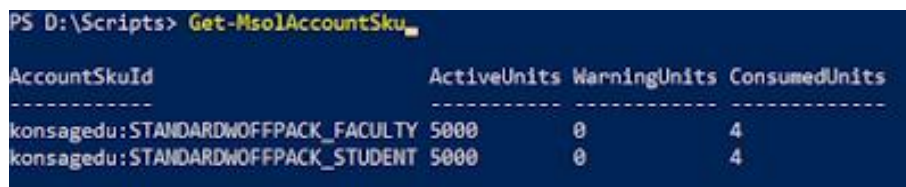
Kilka typowych wyzwań, przed którym staje administrator szkolnej instancji Office 365 to:

1. Zakładanie użytkowników i przypisywanie im licencji .....	2
2. Resetowanie haseł użytkownikom.....	4
3. Tworzenie zespołów Teams .....	5
4. Przypisywanie użytkownikom polis.....	5
5. Zmiana nazwy wyświetlania (Display Name) .....	7

Poniżej przedstawię kilka przykładów, z użyciem klasycznych technik:

## 1. Zakładanie użytkowników i przypisywanie im licencji

Jeżeli szkoła korzysta z Librusa lub Vulcana, to na grupie dostępne są instrukcje, jak zsynchronizować z Office 365 listę nauczycieli/uczniów z użyciem tych systemów. Czasami jednak musimy utworzyć konta dla wielu osób samodzielnie. Najlepiej byłoby (ze względu na stan epidemii i konieczność wdrożenia w jak najkrótszym czasie) utworzyć dla nich konta skryptem i dodatkowo wysłać im loginy i hasła początkowe na już znany adres. Po pierwsze potrzebujemy w tym celu plik csv, z listą użytkowników. Warto pamiętać, że powershell domyślnie akceptuje "," jako znak przystankowy, a dla polskich ustawień regionalnych Microsoft Excel najczęściej tworzy taki plik z polami rozdzielonymi ";" - jest to zaznaczone w przykładowym skrypcie. Żeby konto miało od razu przypisaną licencję, dobrze jest to również załatwić w naszym skrypcie. W pierwszej kolejności potrzebujemy się zalogować do naszej instancji Office 365 (tzw. tenanta) i sprawdzić identyfikatory licencji (niestety w każdej instancji jest dodatkowo doklejany indywidualny identyfikator, czyli mamy taką zbitkę - **konsagedu:STANDARDWOFFPACK\_STUDENT**. W tym wypadku konsagedu jest identyfikatorem mojej testowej organizacji Office 365 (jak widać na obrazku).



```
PS D:\Scripts> Get-MsolAccountSku
```

AccountSkuId	ActiveUnits	WarningUnits	ConsumedUnits
konsagedu:STANDARDWOFFPACK_FACULTY	5000	0	4
konsagedu:STANDARDWOFFPACK_STUDENT	5000	0	4

Jaką wartość wpisać do skryptu sprawdzamy komendą **Get-MsolAccountSku**. Standardowo powinniśmy mieć przypisaną do naszej organizacji licencję A1 dla nauczycieli i A1 dla uczniów (chyba że aktywowaliśmy inne licencje np. do testów). Oczywiście licencje możemy przypisać później, po utworzeniu grup dla poszczególnych klas i nauczycieli. Przypisanie licencji do grup jest realizowane bardzo prosto w portalu administracyjnym Azure AD w zakładce licencje.

Następnie listę nauczycieli/uczniów wyeksportowaną z zewnętrznego źródła importujemy z pliku csv. Plik powinniśmy uzupełnić o kolumnę UserPrincipalName (nie może być w niej polskich znaków), czyli nazwę logowania użytkownika do naszej organizacji Office. Ze względu na RODO dla uczniów nie powinno to być ImieNazwisko, ponieważ w połączeniu ze szkolną domeną stanowi już dane osobowe. Może to być np. identyfikator z dziennika elektronicznego. Jeżeli chcemy wysłać informacje mailem to musimy również wypełnić kolumnę email, która będzie przypisywana do pola Alternatywny Email w Azure AD. Warto również pamiętać, że jeżeli

chcemy wysyłać maile ze skryptu, to musimy się uwierzytelnić i użyć portu 587. Skrypt będzie wyglądał np. tak:

```
# Podajemy poświadczenia administracyjne i łączymy się z chmurą
$cred = get-credential
Connect-MsolService -Credential $cred

# podajemy nazwę licencji dla ucznia
$StudentsLicense = 'konsagedu:STANDARDWOFFPACK_STUDENT'

# importujemy listę uczniów z pliku csv w postaci
# UserPrincipalName;DisplayName;Imie;Nazwisko;email;telefon
# Musimy wskazać znak rozdzielający, jeżeli plik csv ma kolumny
# rozdzielone
# innym znakiem niż ,

$AllOffice365Students = Import-CSV D:\Scripts\naglowek.csv `
-Delimiter ";"

# dla każdego wiersza z pliku csv wykonujemy akcję
foreach ($Student in $AllOffice365Students) {
    # tworzymy nowe konto w usłudze Office 365
    $O365student = New-MsolUser -UserPrincipalName `
    $Student.UserPrincipalName -AlternateEmailAddresses `
    $Student.email -DisplayName $Student.DisplayName `
    -FirstName $Student.Imie -LastName $Student.Nazwisko `
    -mobilePhone $Student.telefon -UsageLocation PL `
    -PreferredLanguage PL-pl -PasswordNeverExpires $true `
    -StrongPasswordRequired $true
    # przypisujemy licencję
    Set-MsolUserLicense -UserPrincipalName `
    $Student.UserPrincipalName -AddLicenses $Studentslicense

    # pobieramy dane po utworzeniu konta
    $mail = $Student.email
    $pwd = $O365student.Password
    $stu = $Student.UserPrincipalName
    $body = "Utworzono konto w systemie Office 365 o nazwie $stu
    Hasło tymczasowe: $Pwd"
    # wysyłamy maila z hasłem na alternatywny adres użytkownika
    Send-MailMessage -SmtpServer smtp.office365.com -usessl `
    -Credential $cred -Port 587 -To "$mail" -From `
    "admin@konsagedu.onmicrosoft.com" -Subject `
    "Office 365 - nowe konto" -Body $body -Encoding UTF8 `
    -BodyAsHtml
}
```

Kod skryptu dostępny na [moim githubie](#).

Id licencji i konto administratora oczywiście w skrypcie jest powiązanie z moją organizacją testową. Uwaga. Jeżeli dla administratora włączyliśmy uwierzytelnianie wieloskładnikowe (MFA), to do wysłania maila musimy użyć hasła aplikacyjnego, a nie tego, którym się normalnie logujemy. UWAGA: od początku marca domyślnie organizacje Office 365 mają włączone tzw. domyślne ustawienia zabezpieczeń, które wymuszają MFA. Jeżeli uznamy, że jest to dla uczniów i nauczycieli zbyt uciążliwe możemy to wyłączyć w portalu administracyjnym Azure AD.

## 2. Resetowanie haseł użytkownikom

W przypadku sytuacji, gdy musimy zresetować hasło grupie użytkowników, możemy postąpić bardzo podobnie - zaimportować listę użytkowników z pliku csv, tym razem wystarczy nam nazwa użytkownika czyli UserPrincipalName i alternatywny adres mailowy. Czyli postępujemy podobnie:

```
# Podajemy poświadczenia administracyjne i łączymy się z chmurą
$cred = get-credential
Connect-MsolService -Credential $cred

$Pwd = 123zxcASD
# importujemy listę uczniów z pliku csv w postaci
# UPN,email

$AllOffice365Students = Import-CSV D:\Scripts\naglowek.csv

# dla każdego wiersza z pliku csv wykonujemy akcję
foreach ($Student in $AllOffice365Students) {
    Set-MsolUserPassword -UserPrincipalName $Student.UPN `
        -NewPassword $Pwd -ForceChangePassword $True

    # pobieramy dane po utworzeniu konta
    $mail = $Student.email
    $stu = $Student.UPN
    $body = "Zresetowano hasło w systemie Office 365. Hasło
tymczasowe: $Pwd"

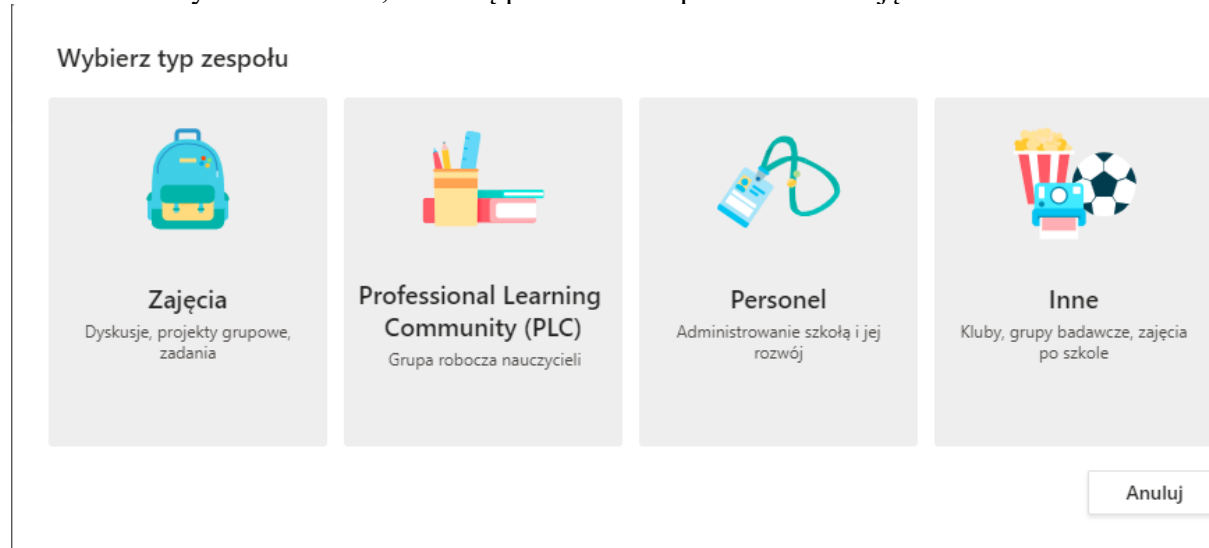
    # wysyłamy maila z hasłem na alternatywny adres użytkownika
    Send-MailMessage -SmtplibServer smtp.office365.com -usessl `
        -Credential $cred -Port 587 -To "$mail" -From `
        "admin@konsagedu.onmicrosoft.com" -Body $body -BodyAsHtml `
        -Subject "Office 365 - zmiana hasła"
}
```

Kod skryptu dostępny na [moim githubie](#).

### 3. Tworzenie zespołów Teams

Jeżeli chodzi o zakładanie zespołów w Teams, to już niedługo będzie to można zrobić z użyciem modułu MicrosoftTeams, jednak jak na razie cmdlet **New-Team** nie obsługuje atrybutu Template, który pozwala nam wybrać szablon klasy lub personelu. Teraz możemy to zrobić z użyciem Graph API. Przykładowy skrypt udostępnili koledzy na githubie grupy Microsoft 365 User Group Poland - <https://github.com/Microsoft-365-User-Group-Poland/Skrypty>.

Warto pamiętać, że standardowo dla zespołów klasowych zakładamy zespół z aplikacji Teams, wybierając szablon Zajęcia (w wersji angielskiej aplikacji – Class). Tylko taki szablon tworzy Notes Klasowy oraz Zadania, które są potrzebne do prowadzenia zajęć.



### 4. Przypisywanie użytkownikom polis

Przypisywanie użytkownikom polis to temat na osobny artykuł - w zależności od obszaru, dla którego polisy mają być użyte, to realizujemy przypisywanie z użyciem różnych modułów:

- polisy związane z hasłami - moduł AzureAD.
- polisy związane ze skrzynką pocztową - moduł Exchange Online (lub nowy moduł ExchangeOnlineManagement).
- polisy dotyczące Teams - moduł MicrosoftTeams (choć niektóre polisy są konfigurowane w module Skype for Business Online), etc.

Myślę, że ciekawym przypadkiem przypisywania polis może być ograniczanie uprawnień uczniom w Teams, poprzez tzw. [Policy Packages](#). W tym celu musimy odfiltrować uczniów, czyli konta z przypisaną licencją ucznia. W tym celu dla odmiany użyję modułu AzureAD. Wyświetlę licencje dostępne w moim tenancie EDU i odfiltruję użytkowników z przypisaną taką licencją. Id licencji sprawdzę komendą get-AzureAdSubscribedSku:

```
PS D:\Scripts> get-AzureAdSubscribedSku | Select-Object -Property SkuPartNumber,SkuId

SkuPartNumber      SkuId
-----
STANDARDWOFFPACK_FACULTY 94763226-9b3c-4e75-a931-5c89701abe66
STANDARDWOFFPACK_STUDENT 314c4481-f395-4525-be8b-2ec4bb1e9d91
```

Mając ID mogę użyć filtra przygotowanego z jego pomocą

```
$students = Get-AzureADUser -All $true | Where-Object
{($_.assignedLicenses).SkuId `
-contains "314c4481-f395-4525-be8b-2ec4bb1e9d91"}
```

Teraz jeszcze tylko sprawdzę, jakie pakiety polis są dostępne w moim testowym tenancie

```
PS D:\Scripts> Get-CsPolicyPackage_

Name      : Education_HigherEducationStudent
Description : This is an Education_HigherEducationStudent package

Name      : Education_PrimaryStudent
Description : This is an Education_PrimaryStudent package

Name      : Education_SecondaryStudent
Description : This is an Education_SecondaryStudent package

Name      : Education_Teacher
Description : This is an Education_Teacher package
```

I można już uruchomić w pętli przypisanie odpowiedniej paczki (np. dla licealistów będzie to Education\_SecondaryStudent).

```
foreach ($student in $students) {Grant-CsUserPolicyPackage `
-Identity $student.UserPrincipalName `
-PackageName Education_SecondaryStudent }
```

Taką samą pętlą możemy sprawdzić, czy polisy package został przypisany poprawnie:

```
foreach ($student in $students) {Get-CsUserPolicyPackage -Identity
$student.UserPrincipalName }
```

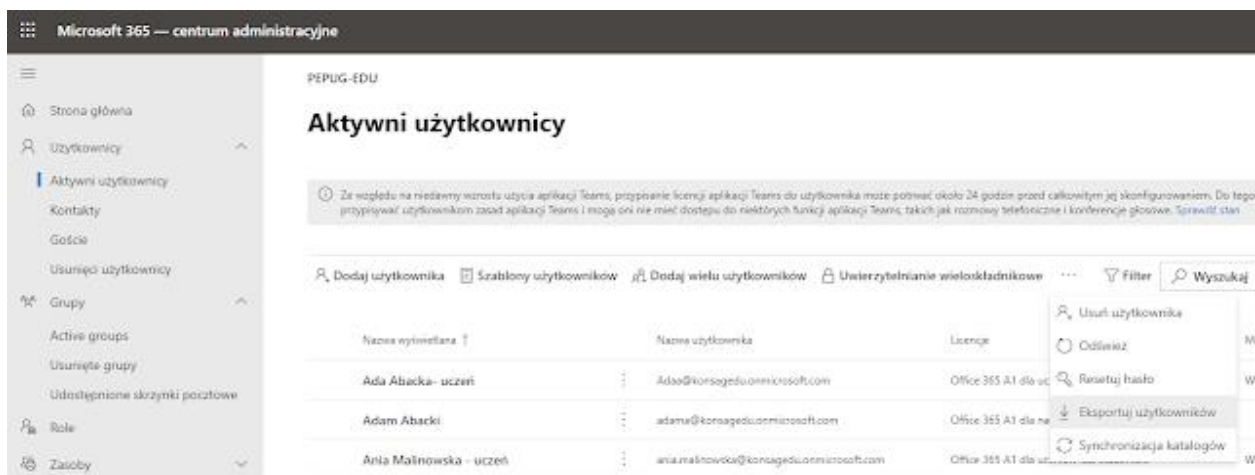
Oczywiście ustawienia polis w ramach pakietów możemy modyfikować bądź też tworzyć własne. UPDATE:

W dokumentacji Microsoft Teams dla EDU pojawił się [rozbudowany artykuł](#) z przykładami przypisywania polis, odpinania polis i powiązanych operacji. Warto przeczytać jako uzupełnienie tego artykułu.



## 5. Zmiana nazwy wyświetlania (Display Name)

Kolejne zagadnienie, to zmiana sortowania użytkowników w zespołach. Niestety domyślnie pole Display Name, które jest wykorzystywane w sortowaniu użytkowników jest tworzone na podstawie pól Imię i Nazwisko. Jednak administrator może zmienić wartość tego pola bez wpływu na działanie systemu. Można edytować każdego użytkownika indywidualnie, ale oczywiście prościej to zrobić z poziomu Powershell. W tym celu możemy użyć pliku csv, użytego do importu danych uczniów i nauczycieli, lub dowolnego innego pliku csv, np. lista użytkowników wyeksportowana z panelu administracyjnego.



W tym celu możemy użyć prostego skryptu, który zaimportuje listę użytkowników z pliku i utworzy pole displayname z połączenia wartości pól z imieniem i nazwiskiem. W tym celu powinniśmy poniższy kod zapisać jako plik z rozszerzeniem ps1 (na przykład zmiana.ps1 w katalogu c:\Scripts) i uruchomić go z Powershella:

```
Import-Module Msonline
Connect-MsolService
#
# nagłówek pliku csv -> UserPrincipalName, Imie, Nazwisko
# jeżeli kolumny rozdzielone są znakiem ";" zamiast "," należy
# użyć przy imporcie atrybutu -Delimiter (import-csv domyślnie szuka ",")
#
$AllStudents = Import-CSV D:\Scripts\uczniowie.csv -Delimiter ";"
ForEach ($student in $AllStudents)
{
    $DisplayName = $student.Nazwisko+" "+$student.Imie
    $currentstudent = get-MsolUser -UserPrincipalName $student.UserPrincipalName
    set-MsolUser -ObjectId $currentstudent.ObjectId -Displayname $DisplayName
}
```

Kod skryptu dostępny również na [moim githubie](#). Możemy również bezpośrednio przetworzyć atrybuty obiektów w naszej organizacji, bez użycia pliku csv wykonując nieco zmodyfikowany skrypt:

```
Import-Module Msonline
Connect-MsolService
```

```
$Allusers = get-MsolUser
ForEach ($user in $Allusers)
{
    $DisplayName = $user.LastName+" "+$user.FirstName
    Set-MsolUser -ObjectId $user.ObjectId -Displayname $DisplayName
}
```

Oczywiście najpierw powinniśmy zainstalować odpowiedni moduł do zarządzania Office 365 (informacja na początku artykułu). Do uruchomienia powyższego skryptu potrzebny jest nam moduł MSOnline, który instalujemy komendą `install-module MSOnline`.