Entra IDをコマンドで操作する方法

～Graph APIを使ったPowerShellでのコマンド管理入門編～

目次

【Chapter1】なぜコマンドで管理するのか

【Chapter1】なぜコマンドで管理するのか

コマンドは一定のルールを把握していれば非常に便利なツールです。
全て覚えておく必要はなく、ChatGPTやCopilotなどの生成AIで作成・修正もできます。

コマンドで管理する

メリット

高効率性

・**手順を**GUIに依存せず**楽に作成・管理**ができる
・複数タスクを一度に自動化できるため、**大量の処理などを高速化できる**

一貫性

同じ作業を繰り返し処理できるため、ヒューマンエラーを減らすことができる

詳細な制御

GUIでは提供されないような詳細オプションなどもあり、より**細かい制御が可能**になる

デメリット

想定しない影響

過剰な権限を与えると、コマンド入力ミスにより**重大な影響を与えるリスク**がある

検証大事

心理的ハードル

・操作の実態が分からない
・コマンドを覚えていないため、どこから手を付ければよいかわからない

大量処理向き

小規模の作業はGUIでやった方が早い場合もある

全てをコマンドで実施できるようにならなくてOK
基本はGUIで操作し、大量作業で負担が大きいときにピンポイントでコマンド使うと楽になるという事を知っておいてください。

【Chapter1】利用シーン

実際に私がコマンドを使って運用・作業したシーンを紹介します。

・3つの検証テナントに同じユーザー登録をする

・100ユーザーに対して「姓」「名」の情報を追加する

・万を超えるユーザーに対してライセンスの特定のアプリケーションを無効化する

・M365の設定投入をコマンドで手順化

など

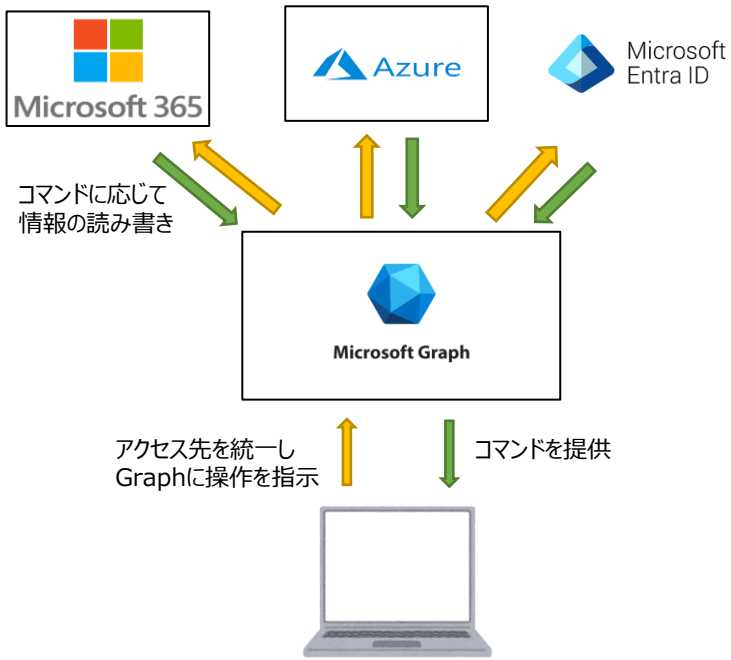
【Chapter2】Microsoft Graphについて

【Chapter2】Microsoft Graphについて

PowerShellでEntra IDアカウントの操作を行うために、PowerShell上でMicrosoft Graphというサービスのコンポーネントを利用します。

＜**Graphとは**＞
Microsoftが提供するクラウドサービスやデータにユーザーがアクセスする際の**APIを提供**します。
Graphは**ユーザーに代わって**様々な**管理センターにアクセス・操作**を実行します。
Graph APIはユーザー管理以外にも、様々な観点での活用ができます。

一元管理	Entra ID以外にも、IntuneやSharePoint、Azureなど各種Microsoftサービスをを Graph APIで一元的に管理ができる
自動化	ログ取得やレポート生成など繰り返し行う作業をスクリプトにすることで自動化できる



＜**委任権限について**＞
Graphはユーザーに代わってテナントの操作を行うため、Graphがどの範囲の操作まで行っていいかを
ユーザーが指定する必要があります。
この委任権限は、以下のように指定します。
Connect-MgGraph -Scopes “XXX.Read.All”,”YYY.ReadWrite.All”

Ex）ユーザー情報に対する読み取り書き込みだけ許可、ライセンス情報に対する読み取りだけ許可　など

一例
ユーザプロパティを読み取って更新をしたい→ "User.ReadWrite.All"
グループの作成をしたい→“Group.ReadWrite.All”
ライセンス適用のとき→“Organization.Read.All”
→だいたい生成AIが教えてくれる

【Chapter3】Graph APIコマンドの共通項目

【Chapter3】共通項目

PowerShellを使ったGraph API操作する上で、全ての操作に共通して必要なコマンドを紹介します

1．PowerShell7以上をインストール&管理者権限で実行

Windowsに初期で入っているPowerShellはバージョン5
MS Storeなどからバージョン7のダウンロードが必要

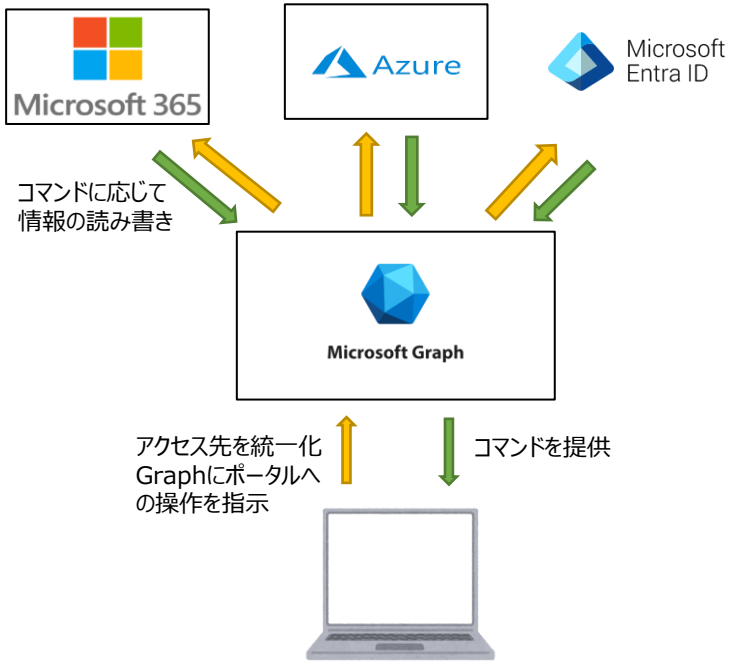
2．以下のコマンドを実行

//Graphのモジュールを設定するPCにインストールし、Graphを使えるようにする。利用端末につき1回限り実施が必要（PowerShellを管理者権限で実行する必要あり）
① Install -module Microsoft.Graph

//Graphに適切な委任権限を付与する。PowerShellを閉じるなど、セッションが切れる毎に実施が必要
② Connect-MgGraph –Scope ‘XXX.ReadWrite.All’

//インストールしたGraphのユーザ処理に関するモジュールをPowerShellにインポートする。
PowerShellを閉じるなど、セッションが切れる毎に実施が必要
③ Import -module Microsoft.Graph

＜参考＞モジュールの更新は「Update-Module Microsoft.Graph 」で実施可能
https://learn.microsoft.com/ja-jp/powershell/microsoftgraph/installation?view=graph-powershell-1.0&toc=%2Fgraph%2Ftoc.json#updating-the-sdk



動詞	意味
Get	テナントから情報を取得する
Set	情報を設定する
Update	情報を更新する
New	新規で作成する
Remove	削除する

＜文法＞
動詞＋対象＋オプション（New-MgUser -parameter や Remove-MgGroup -pamaeter など）

【Chapter4】ユーザー作成

【Chapter4】ユーザー作成

ユーザー作成を実行するためのコマンドと、必要なパラメータについて解説します。

ユーザ作成のコマンド
> New-MgUser –displayName $DisplayName ~~

＜入力パラメータ＞

種別	パラメータ名	必須	記載方法
ポータル上での表示名	displayName	○	
ユーザアカウント	userPrincipalName	○	xxx@yyy.onmicrosoft.com
アカウントの有効化	accountEnabled	○	bool型、Trueで有効Falseで無効
電子メールのエイリアス	mailNickname	○	
パスワードのプロファイル	passwordProfile	△	**forceChangePasswordNextSignIn**： 次のサインイン時にパスワード強制変更。Bool **forceChangePasswordNextSignInWithMfa**： 次のサインイン時にパスワード強制変更＋MFAの設定 **password**： ユーザのパスワード
苗字	givenName	×	
名前	surname	×	
利用場所	usageLocation	△	日本はJP。ライセンスを付与するためには必須

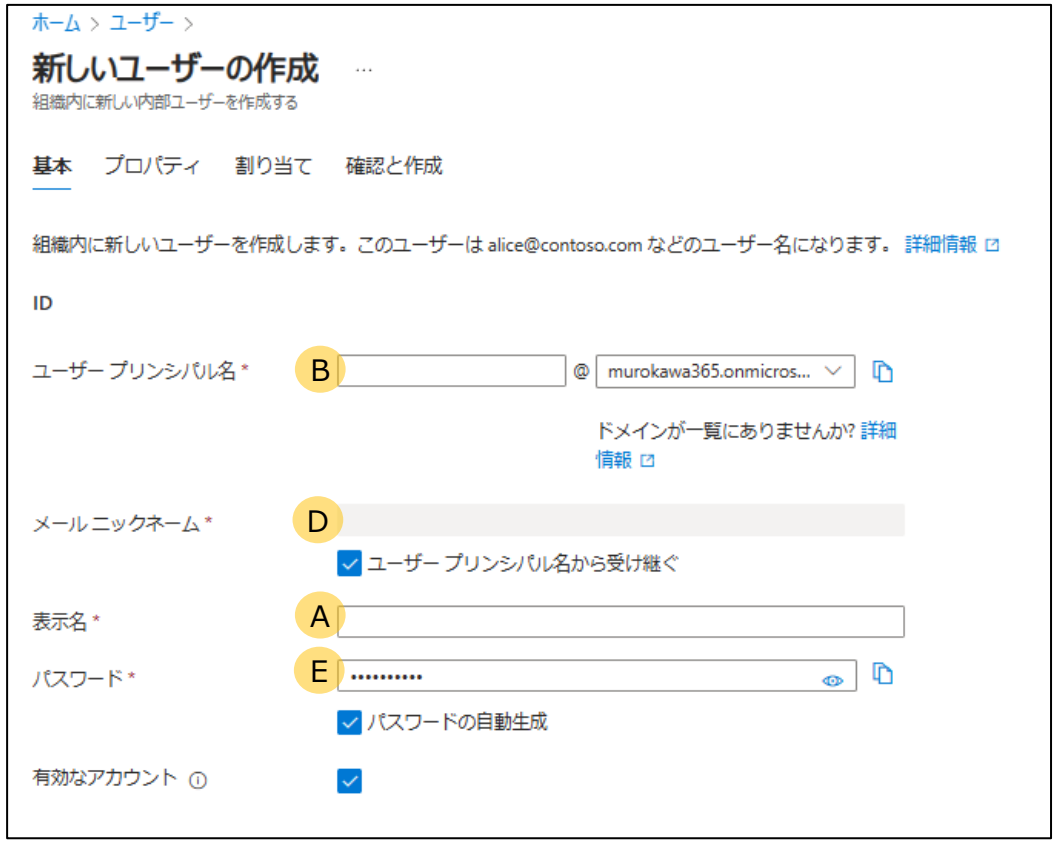
【Chapter4】ユーザー作成

<ユーザー作成のためのCSVの作成例>

必須項目以外は
必要な列だけでOK

	A	B	C	D	E	F	G	H	I	J
1	displayNa	userPrinci	accountEr	mailNickn	password	forceChan	givenNam	surname	usageLocation	
2	ユーザー1	user1@wv	TRUE	user1	Q9b3ju5t	TRUE	情報	太郎	JP	
3	ユーザー2	user2@wv	TRUE	user2	Q9b3ju5t	TRUE	栄愛	花子	JP	
4	ユーザー3	user3@wv	TRUE	user3	Q9b3ju5t	TRUE	妻夫木	聡	JP	

<ポータル画面で作成する場合との比較>



【Chapter4】ユーザー作成

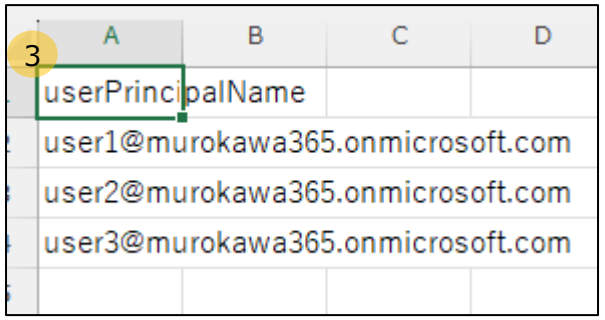
```
//ユーザー作成の委託権限はUser.ReadWrite.All
Connect-MgGraph -Scopes "User.ReadWrite.All"

Import-Module -Name Microsoft.Graph.Users

//csvという変数にCSVファイルの中身を格納
$csv = Import-Csv -Path "C:XXX¥UserCreateTemplate.csv"

//CSVを1行ずつ処理するためにforeach文を使用。eachlineという任意の変数にcsv変数を1行ずつ入れていく。
foreach($eachline in $csv){

//paramsという変数にパラメータを格納していく
   $params = @{
      DisplayName = $eachline.displayName
      UserPrincipalName = $eachline.userPrincipalName
      PasswordProfile = @{
         Password = $eachline.password
               //csvはデフォルトでStringなので、Bool値で入れたいときは以下のようにコンバートする必要がある。
         ForceChangePasswordNextSignIn = [System.Convert]::ToBoolean($eachline.forceChangePasswordNextSignIn)
      }
      AccountEnabled = [System.Convert]::ToBoolean($eachline.accountEnabled)
      MailNickname = $eachline.mailNickname
      GivenName = $eachline.givenName
      Surname = $eachline.surname
      UsageLocation = $eachline.usageLocation
   }
   New-MgUser -BodyParameter $params
}
```

【Chapter5】グループへのメンバー追加

【Chapter5】グループへのメンバー追加

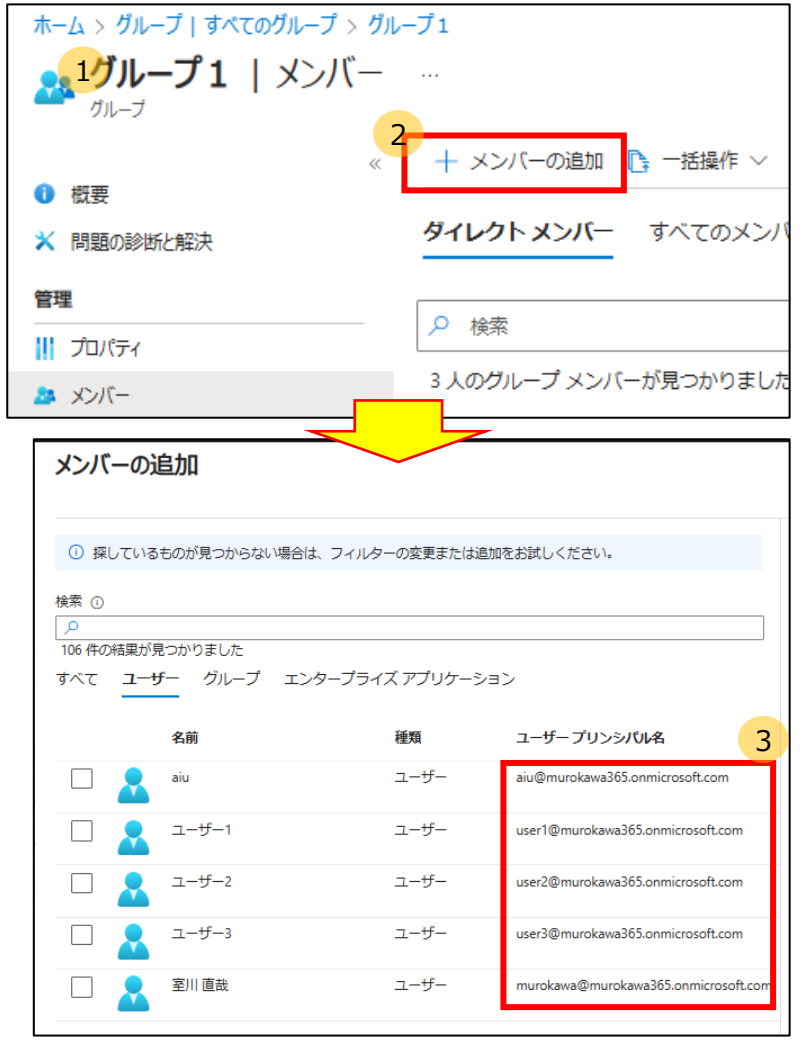
グループへのメンバー追加は、UPNを指定して追加したいグループIDのメンバーに加えます。

＜グループメンバー追加のためのCSVの作成例＞




```
//グループのメンバーシップ管理委託権限はUser.ReadWrite.Allと、ロール割り当て可能なグループの場合、
  RoleManagement.ReadWrite.Allも必要
Connect-MgGraph -Scopes "User.ReadWrite.All", "RoleManagement.ReadWrite.Directory"

Import-Module -Name Microsoft.Graph

//csvという変数にCSVファイルの中身を格納
$csv = Import-Csv -Path "C:XXX¥GroupMemberAdd.csv"   3

//グループの情報を取得
$groupId = (Get-MgGroup -Filter “displayName eq ‘追加したいグループ名'").Id   1

//グループをユーザーに追加
foreach ($eachline in $csv) {
        $userId = (Get-MgUser -Filter "userPrincipalName eq '$($eachline.userPrincipalName)'").Id
    2 New-MgGroupMember -GroupId $groupId -DirectoryObjectId $userId
}
```

＜グループメンバー追加のポータル画面＞



【Chapter6】ライセンス割り当て

【Chapter6】ライセンス割り当て

ライセンス付与は、付与したいUPNにライセンスのIDを指定して割り当てます。

＜ライセンス付与のためのCSVの作成例＞

	A	B	C	D
1	2 userPrin 1	icenseSKU		
2	user1@wv	DEVELOPERPACK_E5		
3	user2@wv	DEVELOPERPACK_E5		
4	user3@wv	DEVELOPERPACK_E5		
5				

//Connectの委任権限にOrganization.Read.Allを追加　※リソースの読み取り実行権限。

```
Connect-MgGraph -Scopes "Organization.Read.All", "User.ReadWrite.All"

Import-Module -Name Microsoft.Graph.Users.Actions
Get-MgSubscribedSku | Select-Object SkuPartNumber, SkuId, ServiceInfo

$csv = Import-Csv “C:XXX¥licenseAssignmentsUser.csv“
```

//変数にライセンス情報を格納　1 2

```
foreach ($eachline in $csv) {
$e5Sku = Get-MgSubscribedSku -All | Where-Object { $_.SkuPartNumber -eq $eachline.licenseSKU }
```

//格納されてるか一応確認

```
$e5Sku.SkuId
```

※結果が無い場合はライセンス名が違う可能性がある

```
 $allSkus | Format-Table SkuPartNumber, SkuId, ConsumedUnits, PrepaidUnits
```

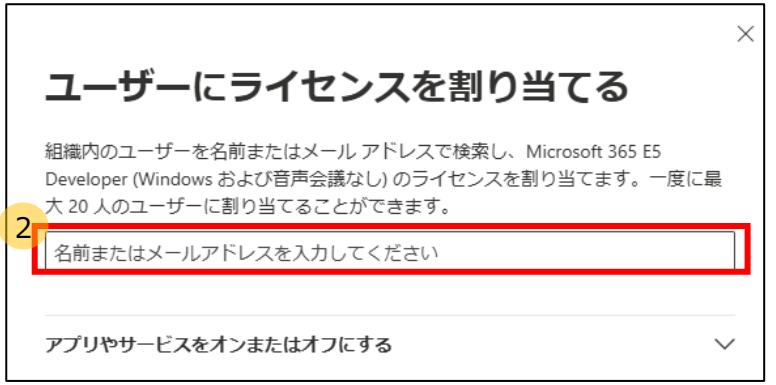
で情報を表示

//UPNを指定してライセンス適用

```
$skuId = $e5Sku.SkuId
Set-MgUserLicense -UserId $eachline.userPrincipalName -AddLicenses @{SkuId = $skuId} -RemoveLicenses @()
}
```

2

＜ライセンス付与のポータル画面＞



【Chapter6】ライセンス割り当て

＜主要なSkuPartNumber一覧＞

ライセンス名	SkuPartNumber	ライセンス名	SkuPartNumber
Microsoft Entra Basic	AAD_BASIC	**Microsoft 365 Business Basic**	O365_BUSINESS_ESSENTIALS
Microsoft Entra ID P1	AAD_PREMIUM	**Microsoft 365 Business Standard**	O365_BUSINESS_PREMIUM
Microsoft Entra ID P2	AAD_PREMIUM_P2	**Microsoft 365 E3**	SPE_E3
Enterprise Mobility + Security E3	EMS	**Microsoft 365 E5**	SPE_E5
Enterprise Mobility + Security E5	EMSPREMIUM	**Microsoft 365 E5 （CDX試用版）**	Microsoft_365_E5_(no_Teams)
Microsoft 365 A1	M365EDU_A1		

その他
https://learn.microsoft.com/ja-jp/entra/identity/users/licensing-service-plan-reference

【Chapter7】まとめ

【Chapter7】まとめ

本勉強会ではコマンドを使ったユーザー管理の方法について解説しました。
コマンドは数が多く内容を網羅するのは難しいため、AIとラリーしながらドキュメントを読んでいくことをお勧めします。

コマンド管理について（まとめ）

・全てをコマンドで実施できることが大切ではない。状況に合わせて、1つの便利ツールとして認識することが大切
・ポータル上でも同様の操作をできるようにしておくことで、コマンド利用イメージの具体化や、ポータルorコマンドどっちで作業するの判断が付きやすいので、
ポータル操作も一つのスキルとして意識する
・GUIほどの頻度ではないがコマンドも変更があることや、コピペミス等により意図しない影響を及ぼす可能性があるため、**本番環境前にテストを行う**

本日の勉強会は以上で終了です。
ご清聴ありがとうございました。