Create app - CSP

Wednesday, August 7, 2019 9:05 AM

```
<#
    .SYNOPSIS
        This script will create the require Azure AD application.
    .EXAMPLE
        .\Create-AzureADApplication.ps1 -ConfigurePreconsent -DisplayName "Partner Center Web App"
.\Create-AzureADApplication.ps1 -ConfigurePreconsent -DisplayName "Partner Center Web App" -TenantId
eb210c1e-b697-4c06-b4e3-8b104c226b9a
.\Create-AzureADApplication.ps1 -ConfigurePreconsent -DisplayName "Partner Center Web App" -TenantId
tenant01.onmicrosoft.com
    .PARAMETER ConfigurePreconsent
        Flag indicating whether or not the Azure AD application should be configured for preconsent.
    .PARAMETER DisplayName
        Display name for the Azure AD application that will be created.
    .PARAMETER TenantId
        [OPTIONAL] The domain or tenant identifier for the Azure AD tenant that should be utilized to create the
various resources.
#>
Param
(
    [Parameter(Mandatory = $true)]
    [switch]$ConfigurePreconsent,
    [Parameter(Mandatory = $true)]
    [string]$DisplayName,
    [Parameter(Mandatory = $false)]
    [string]$TenantId
)
$ErrorActionPreference = "Stop"
# Check if the Azure AD PowerShell module has already been loaded.
if ( ! ( Get-Module AzureAD ) ) {
    # Check if the Azure AD PowerShell module is installed.
    if ( Get-Module -ListAvailable -Name AzureAD ) {
        # The Azure AD PowerShell module is not load and it is installed. This module
        # must be loaded for other operations performed by this script.
        Write-Host -ForegroundColor Green "Loading the Azure AD PowerShell module..."
        Import-Module AzureAD
    } else {
        Install-Module AzureAD
    }
}
try {
    Write-Host -ForegroundColor Green "When prompted please enter the appropriate credentials..."
if([string]::IsNullOrEmpty($TenantId)) {
        Connect-AzureAD | Out-Null
$TenantId = $(Get-AzureADTenantDetail).ObjectId
    } else {
        Connect-AzureAD -TenantId $TenantId | Out-Null
    }
} catch [Microsoft.Azure.Common.Authentication.AadAuthenticationCanceledException] {
    # The authentication attempt was canceled by the end-user. Execution of the script should be halted.
    Write-Host -ForegroundColor Yellow "The authentication attempt was canceled. Execution of the script will
be halted..."
    Exit
} catch {
    # An unexpected error has occurred. The end-user should be notified so that the appropriate action can be
```

```
taken.
    Write-Error "An unexpected error has occurred. Please review the following error message and try again." `
      "$($Error[0].Exception)"
}
$adAppAccess = [Microsoft.Open.AzureAD.Model.RequiredResourceAccess]@{
    ResourceAppId = "00000002-0000-0000-c000-000000000000";
    ResourceAccess =
    [Microsoft.Open.AzureAD.Model.ResourceAccess]@{
        Id = "5778995a-e1bf-45b8-affa-663a9f3f4d04";
        Type = "Role"},
    [Microsoft.Open.AzureAD.Model.ResourceAccess]@{
        Id = "a42657d6-7f20-40e3-b6f0-cee03008a62a";
        Type = "Scope"},
    [Microsoft.Open.AzureAD.Model.ResourceAccess]@{
        Id = "311a71cc-e848-46a1-bdf8-97ff7156d8e6";
        Type = "Scope"}
}
$graphAppAccess = [Microsoft.Open.AzureAD.Model.RequiredResourceAccess]@{
    ResourceAppId = "00000003-0000-0000-c000-000000000000";
    ResourceAccess =
        [Microsoft.Open.AzureAD.Model.ResourceAccess]@{
            Id = "bf394140-e372-4bf9-a898-299cfc7564e5";
            Type = "Role"},
        [Microsoft.Open.AzureAD.Model.ResourceAccess]@{
            Id = "7ab1d382-f21e-4acd-a863-ba3e13f7da61";
            Type = "Role"}
}
$partnerCenterAppAccess = [Microsoft.Open.AzureAD.Model.RequiredResourceAccess]@{
    ResourceAppId = "fa3d9a0c-3fb0-42cc-9193-47c7ecd2edbd";
    ResourceAccess =
        [Microsoft.Open.AzureAD.Model.ResourceAccess]@{
            Id = "1cebfa2a-fb4d-419e-b5f9-839b4383e05a";
            Type = "Scope"}
}
$SessionInfo = Get-AzureADCurrentSessionInfo
Write-Host -ForegroundColor Green "Creating the Azure AD application and related resources..."
$app = New-AzureADApplication -AvailableToOtherTenants $true -DisplayName $DisplayName -IdentifierUris
"https://$($SessionInfo.TenantDomain)/$((New-Guid).ToString())" -RequiredResourceAccess $adAppAccess,
$graphAppAccess, $partnerCenterAppAccess -ReplyUrls @("urn:ietf:wg:oauth:2.0:oob")
$password = New-AzureADApplicationPasswordCredential -ObjectId $app.ObjectId
$spn = New-AzureADServicePrincipal -AppId $app.AppId -DisplayName $DisplayName
if($ConfigurePreconsent) {
    $adminAgentsGroup = Get-AzureADGroup -Filter "DisplayName eq 'AdminAgents'"
    Add-AzureADGroupMember -ObjectId $adminAgentsGroup.ObjectId -RefObjectId $spn.ObjectId
}
Write-Host "ApplicationId      = $($app.AppId)"
Write-Host "ApplicationSecret   = $($password.Value)"
```

Performing the Consent

Next you will need to invoke the New-PartnerAccessToken command as shown below to perform the consent process.

PowerShellCopy

```
$credential = Get-Credential
$token = New-PartnerAccessToken -Consent -Credential $credential -Resource
```

```
https://api.partnercenter.microsoft.com -TenantId '<Your Tenant Id>'
```

When the command is invoked, you will be prompted to enter a username and password. Specify the application identifier as the username and the application secret as the password. When then New-PartnerAccessToken command is invoked, you will be prompted for credentials once again. This time you will need to specify the credentials for the service account that you will be using. The service account should be a partner account with the appropriate permissions. After the successful execution of the command, you will find that the `$token` variable contains the response from Azure Active Directory for a token. In the response is a refresh token, you will want to store this value in a secure repository such as Azure Key Vault or a similar service.

Using the Refresh Token

Using the Connect-PartnerCenter command you can connect to Partner Center. You will need to obtain the refresh token value from the secure repository where you stored it. Execute the following commands to request an access token and use it when connecting to Partner Center.

PowerShellCopy

```
$refreshToken = 'Enter the refresh token value here'
$credential = Get-Credential
$pcToken = New-PartnerAccessToken -RefreshToken $refreshToken -Resource
https://api.partnercenter.microsoft.com -Credential $credential
$tenantId = '<Your Tenant Id>'
Connect-PartnerCenter -AccessToken $pcToken.AccessToken -ApplicationId $appId -TenantId $tenantId
```

When you are prompted for credentials specify the application identifier and application secret, for the Azure AD application used when generating the refresh token.

Azure

The Az and Azure PowerShell modules both support the ability to authenticate using access tokens. The following commands demonstrate hot utilize a refresh token to obtain the required access token to connect using the Connect-AzAccount or Connect-AzureRmAccount commands.

PowerShellCopy

```
$credential = Get-Credential
$refreshToken = 'Your-Refresh-Token-Value'
$azureToken = New-PartnerAccessToken -RefreshToken $refreshToken -Resource
https://management.azure.com/ -Credential $credential
$graphToken =  New-PartnerAccessToken -RefreshToken $refreshToken -Resource
https://graph.microsoft.com -Credential $credential
# Az Module
Connect-AzAccount -AccessToken $azureToken.AccessToken -GraphAccessToken $graphToken.AccessToken -
TenantId '<TenantId>'
# AzureRM Module
```

```
Connect-AzureRmAccount -AccessToken $azureToken.AccessToken -GraphAccessToken
$graphToken.AccessToken -TenantId '<TenantId>'
```

MSOnline

The MSOnline PowerShell module support authentication using access tokens. The following commands demonstrate how to utilize the refresh token to obtain the required access token to connect using the Connect-MsolService command.

PowerShellCopy

```
$credential = Get-Credential
$refreshToken = 'Your-Refresh-Token-Value'
$aadGraphToken = New-PartnerAccessToken -RefreshToken $refreshToken -Resource
https://graph.windows.net -Credential $credential
$graphToken =  New-PartnerAccessToken -RefreshToken $refreshToken -Resource
https://graph.microsoft.com -Credential $credential
Connect-MsolService -AdGraphAccessToken $aadGraphToken.AccessToken -MsGraphAccessToken
$graphToken.AccessToken
```