Partner Center: Secure application model

October 2018

Microsoft

Contents

Overview

Microsoft is introducing a secure, scalable framework for authenticating cloud solution provider (CSP) partners and control panel vendors (CPV) through the Microsoft Azure multi-factor authentication (MFA) architecture. CSP partners as well as control panel vendors can rely on the new model to elevate security for Partner Center API integration calls. This will help all parties including Microsoft, CSP partners and control panel vendors to protect their infrastructure and customer data from security risks.

Scope

This document covers the following actors:

- Control panel vendors (CPV)* - A control panel vendor is an independent software vendor that develops apps for use by CSP partners to integrate with Partner Center APIs. A control panel vendor is not a CSP partner with direct access to the Partner dashboard or APIs.
- CSP indirect providers and CSP direct partners who are using app ID + user authentication and directly integrate with Partner Center APIs.

* Perquisite: To qualify as a CPV, you must on board to Microsoft Partner Center as a control panel vendor first, more information to come.

NOTE: If you are an existing CSP partner who is also a CPV, this prerequisite applies to you, as well.

Secure application development guide

Control panel vendors are the companies that develop applications enabling CSPs to sell their products through a unified marketplace. These are usually web applications. In the process of placing orders for Microsoft products on behalf of CSPs, these marketplace applications interact with Microsoft APIs to place orders and provision resources for customers on behalf of CSPs.

Some of these Microsoft APIs include:

- Microsoft Partner Center APIs implementing commerce operations like placing orders and managing subscription lifecycles.
- Microsoft Graph APIs which implement identity management for CSP tenants and CSP customer's tenants.
- Microsoft Azure Resource Manager APIs implementing Azure deployment functionality.

CSP partners are empowered with delegated privileges to act on behalf of their customers when calling Microsoft APIs. Delegated privileges allow CSP partners to complete purchase, deployment, and support scenarios for their customers.

Marketplace applications are designed to help CSP partners list their solutions for customers. To achieve this, marketplace applications need to impersonate CSP partner privileges to call Microsoft APIs. Since CSP partner privileges are very high and provide access to *all* customers of the partner, it is crucial to understand how these applications must be designed to withstand security exploitation vectors. Security attacks on these sensitive applications can lead to the compromise of customer data. Therefore, permission grants and partner privilege impersonation must be designed to follow the principle of least privilege. We will go through principles and best practices to make marketplace applications sustainable and robust from security compromises.

Security principles for credentials impersonation

- Marketplace applications must not store any credentials from CSP partners.
  - CSP partner user passwords should not be shared.
  - CSP partner tenant web app keys must not be shared with control panel vendors.
- A marketplace application must present the application identity along with partner information as opposed to using only partner credentials when making calls impersonating a CSP partner identity.
- Access to a marketplace application must be based on the principle of least privilege and clearly articulated in permissions.
- Authorization for a marketplace application must be pivoted to multiple credentials.
  - Application credentials and partner credentials must be provided together to gain access. There must be no single point of compromise.
  - Access must be restricted to a specific audience or API.
  - Access must identify the purpose of the impersonation.
- Access permissions for a marketplace application must be time bound. CSP partners must be able to renew or revoke access to the marketplace application.
- Quick control or remediation processes must be in place to handle compromises of marketplace application credentials.
- All user accounts should use two-factor authentication (2FA).
- The application model should be friendly to additional security provisions, like conditional access to a better security model.

NOTE: CSP indirect providers and CSP direct partners who are using app ID + user authentication and directly integrate with Partner Center APIs are required to follow the above principles to secure their own marketplace applications.

Application identity and concepts

Multi-tenant applications

A multi-tenant application is generally a Software as a Service (SaaS) application. You can configure your application to accept sign-ins from any Azure Active Directory (Azure AD) tenants by configuring the application type as **multi-tenant** on the Azure dashboard. Users in any Azure AD tenant will be able to sign in to your application after consenting to use their account with your application.

Please see How to sign in any Azure Active Directory user using the multi-tenant application pattern for more information.

Consent framework

For a user to sign in to an application in Azure AD, the application must be represented in the user's tenant. This allows the organization to do things like apply unique policies when users from their tenant sign in to the application. For a single tenant application, this registration is simple; it's the one that happens when you register the application in the Azure dashboard.

For a multi-tenant application, the initial registration for the application lives in the Azure AD tenant used by the developer. When a user from a different tenant signs in to the application for the first time, Azure AD asks them to consent to the permissions requested by the application. If they consent, then a representation of the application called a service principal is created in the user's tenant, and the sign in process can continue. A delegation is also created in the directory that records the user's consent to the application.

NOTE: CSP indirect providers and CSP direct partners who are using app ID + user authentication and directly integrate with Partner Center APIs will have to give consent to their marketplace application using the same consent framework.

This consent experience is affected by the permissions requested by the application. Azure AD supports two kinds of permissions, app-only and delegated.

1. App-only permission is granted directly to the identity of the application. For example, you can grant an application permission to read the list of users in a tenant, regardless of who is signed in to the application.
2. A delegated permission grants an application the ability to act as a signed in user for a subset of the things the user can do. For example, you can grant an application the delegated permission to read the signed in user's calendar.

Some permissions can be consented to by a regular user, while others require a tenant administrator's consent. Please refer to Azure active directory developer documentation for consent framework.

- Scopes, permissions, and consent in the Azure Active Directory v2.0 endpoint
- Understanding user and admin consent

Multi-tenant application open authorization (OAuth) token flow

In a multi-tenant application open authorization (OAuth) flow, the application is represented as a multi-tenant application in the CPV or CSP partner's tenant.

To access Microsoft APIs (Partner Center APIs, Graph APIs, and so on), CSP partners must log in to the application and consent to allow the application to call APIs on their behalf.

> **NOTE**: CSP indirect providers and CSP direct partners who are using app ID + user authentication and directly integrate with Partner Center APIs will have to give consent to their marketplace application using the same consent framework.

The application gains access to the partner's resources, like Graph and Partner Center APIs, through consent and OAuth grants.

Create a multi-tenant application

A multi-tenant application must have the following characteristics:

1. It must be a web app with an application ID and secret key
2. It must have implicit authentication mode turned off.

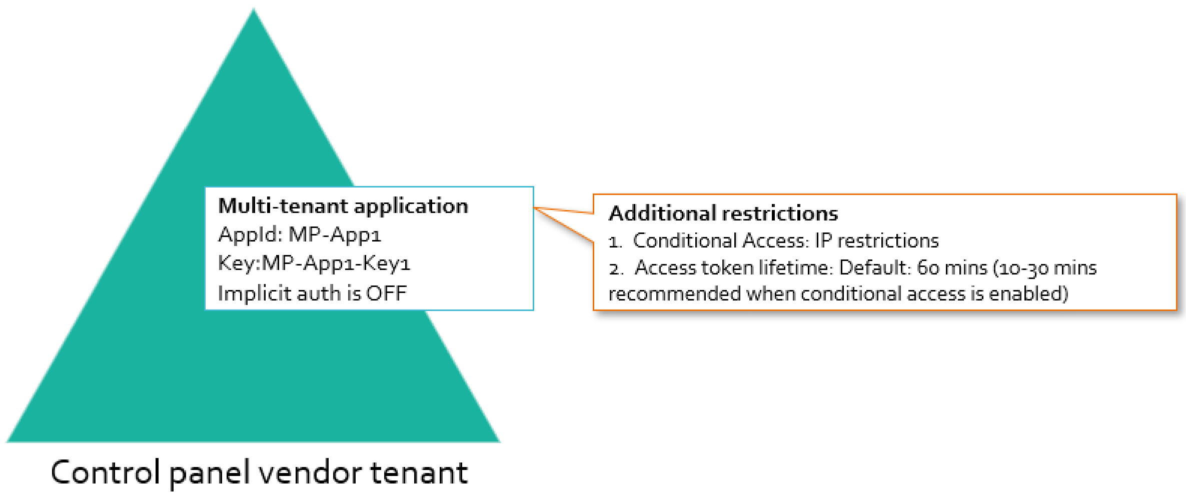
Additional considerations:

- Use a certificate for the secret key.
- Enable conditional access to apply IP range restrictions. This may require additional functionality to be enabled on the Azure AD tenant.
- Apply access token lifetime policies for the application.

While acquiring a token, the app ID and secret key must be presented. The secret key can be a certificate.

The application can be configured to call multiple APIs including Azure Resource Manager APIs. The following are the minimum set of permissions required for Partner Center APIs:

- Azure Active Directory delegated permissions: **Access the directory as signed in user**
- Partner Center APIs delegated permissions: **Access**


Multi-tenant application
AppId: MP-App1
Key:MP-App1-Key1
Implicit auth is OFF
Additional restrictions
1. Conditional Access: IP restrictions
2. Access token lifetime: Default: 60 mins (10-30 mins recommended when conditional access is enabled)
Control panel vendor tenant

1. While acquiring token, the app-id and secret key (can be a certificate) must be presented.
2. The application is configured with specific permissions to access
   1. Audience graph API: read singed in user profile
   2. Partner center APIs: Access

Application captures partner consent

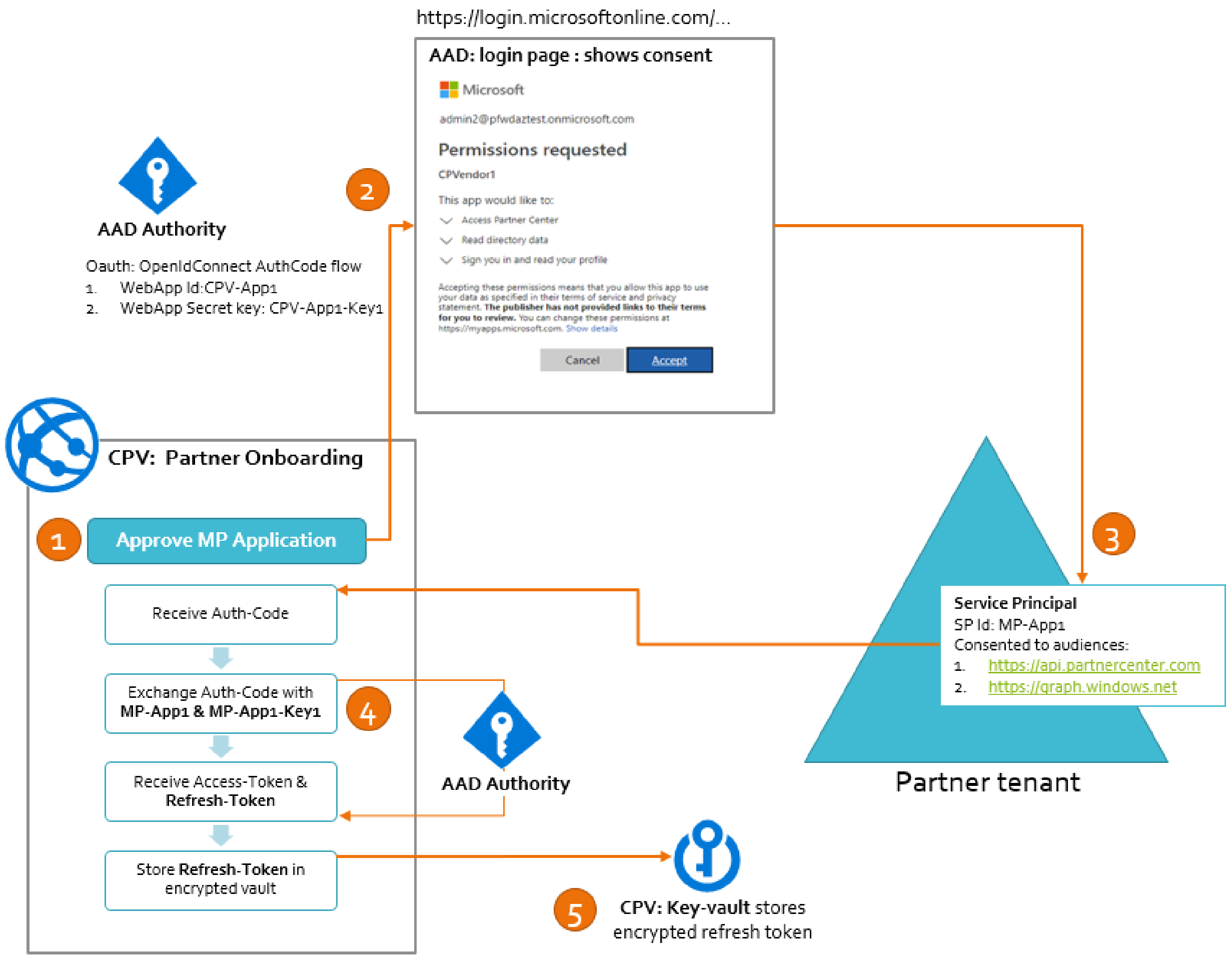
A multi-tenant application must acquire consent from partners and use the consent and grant to make further calls to Partner Center APIs. Consent is acquired through an OAuth authentication code flow. To acquire consent, control panel vendors or CSP partners must build an onboarding website that can accept an authentication code grant from Azure active directory. Please see Authorize access to Azure Active Directory web applications using the OAuth 2.0 code grant flow for more information.

Here are the steps for a multi-tenant application to capture CSP partner consent along with a reusable token for making calls to Partner Center APIs.

Acquiring partner consent

1. Build a partner onboarding web application that can host a consent link for the partner to click through to accept consent for the multi-tenant application.
2. The CSP partner clicks on the consent link. Example link: https://login.microsoftonline.com/common/oauth2/authorize?&client_id=<marketplaceappid>&response_type=code&redirect_url=https://<marketplaceapplicationurl_which_collects_refreshtoken>
3. The Azure AD login page explains the permissions that will be granted to the application on behalf of the user. The CSP partner can decide to use either Admin Agent or Sales Agent credentials to sign in and approve the consent. The application is given permissions based on the user role used to sign in.

4. Once consent is granted, Azure Active Directory creates a service principal of the control panel vendor's multi-tenant application into the CSP partner's tenant. The application is given OAuth grants to act on behalf of the user. These grants allow the multi-tenant application to call Partner Center APIs on behalf of the partner. At this point, the Azure AD login page redirects to the partner onboarding web application. The web application receives an authorization code from Azure AD. The partner onboarding web application must use the authorization code along with the application ID and secret key to call the Azure AD Tokens API to obtain a refresh token.
5. Securely store the refresh token. The refresh token is part of the partner credentials used to obtain access to Partner Center APIs on behalf of the partner. Once the refresh token is acquired, encrypt it and store it in a secret key store, such as the Azure key vault.


https://login.microsoftonline.com/...
AAD: login page : shows consent
Microsoft
admin2@pfwdaztest.onmicrosoft.com
Permissions requested
CPVendor1
This app would like to:
Access Partner Center
Read directory data
Sign you in and read your profile
Cancel
Accept
AAD Authority
Oauth: OpenIdConnect AuthCode flow
1. WebApp Id:CPV-App1
2. WebApp Secret key: CPV-App1-Key1
2
CPV: Partner Onboarding
1
Approve MP Application
Receive Auth-Code
Exchange Auth-Code with MP-App1 & MP-App1-Key1
4
Receive Access-Token & Refresh-Token
Store Refresh-Token in encrypted vault
AAD Authority
3
Service Principal
SP Id: MP-App1
Consented to audiences:
1. https://api.partnercenter.com
2. https://graph.windows.net
Partner tenant
5
CPV: Key-vault stores encrypted refresh token

Accepting these permissions means that you allow this app to use your data as specified in their terms of service and privacy statement. **The publisher has not provided links to their terms for you to review.** You can change these permissions at https://myapps.microsoft.com. Show details

Token request call flow

A control panel vendor's or CSP partner's application must acquire an access token before making calls to Partner Center APIs. The Microsoft Partner Center APIs are represented at resource URL https://api.partnercenter.microsoft.com.

A CPV application should identify which partner account it must impersonate to call Partner Center APIs based on either product or federated sign in. The application retrieves the encrypted refresh token for that partner tenant from the secret key store. The refresh token must be decrypted before use.

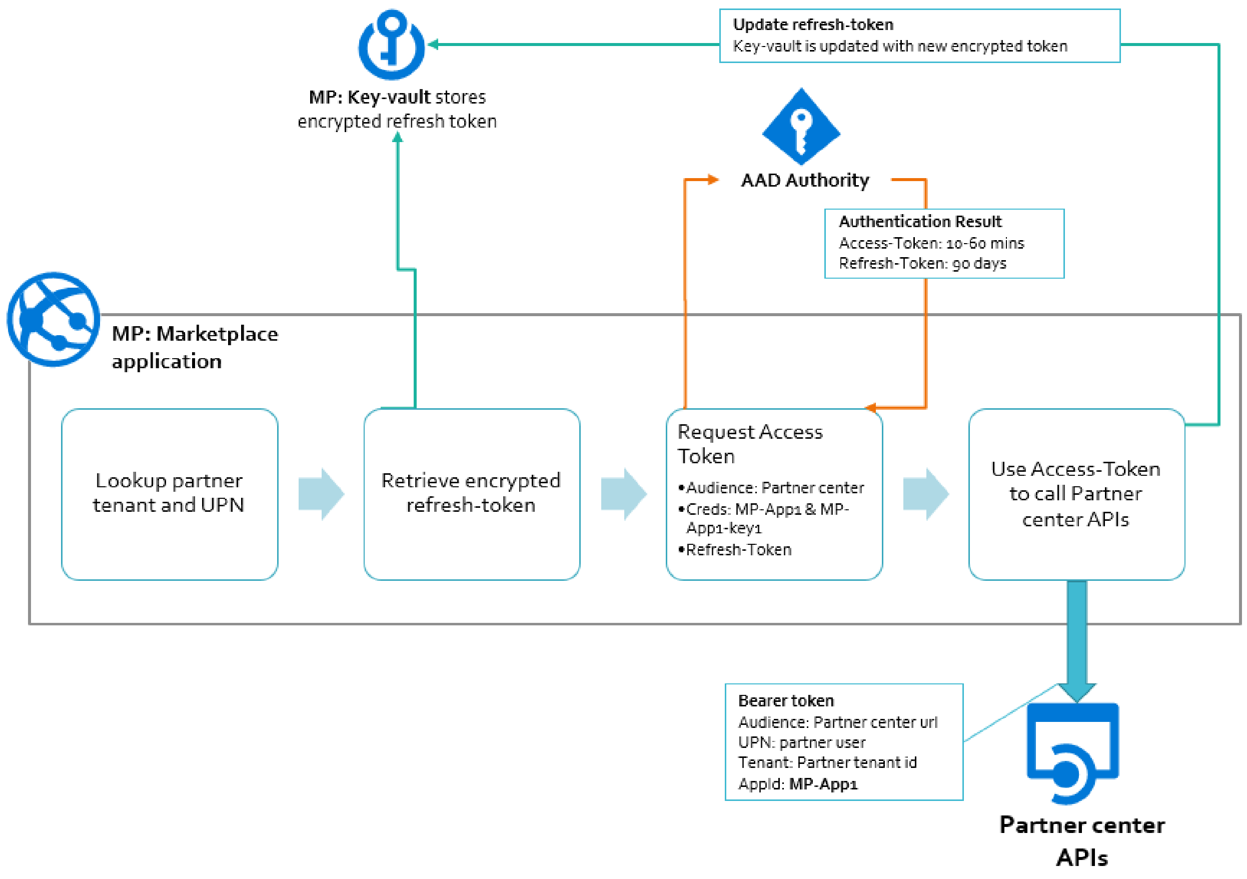
For CSP partners where there is only one tenant who gives consent. The partner account refers to the CSP partner's tenant.

The refresh token is a multi-audience token. That means the refresh token can be used to obtain a token for multiple audiences based on the consent that is granted. For example, if partner consent is given for Microsoft Partner Center APIs and Microsoft Graph APIs, the refresh token can be used to request an access token for both APIs. The access token will have the "on behalf of" grant and will allow a marketplace application to impersonate the partner who consented while calling these APIs.

An access token can be acquired for a single audience at a time. If an application needs to access multiple APIs, it must request multiple access tokens for the targeted audience. To request an access token, the application needs to call the Azure AD Tokens API. Alternatively, it could also use the Azure AD SDK's **AuthenticationContext.AcquireTokenAsync** method, passing in following information:

- *Resource URL*: Endpoint URL for the application to be called. For example, the resource URL for the Microsoft Partner Center API is https://api.partnercenter.microsoft.com.
- *Application Credentials*: The web app's application ID and secret key.
- *The refresh token*

The resulting access token will allow the application to make calls to APIs that are mentioned in the resource. The application cannot request an access token for APIs which were not granted permission as part of the consent request. The UserPrincipalName (UPN) attribute value is the Azure AD username for the user accounts.



Update refresh-token
Key-vault is updated with new encrypted token
MP: Key-vault stores encrypted refresh token
AAD Authority
Authentication Result
Access-Token: 10-60 mins
Refresh-Token: 90 days
MP: Marketplace application
Lookup partner tenant and UPN
Retrieve encrypted refresh-token
Request Access Token
•Audience: Partner center
•Creds: MP-App1 & MP-App1-key1
•Refresh-Token
Use Access-Token to call Partner center APIs
Bearer token
Audience: Partner center url
UPN: partner user
Tenant: Partner tenant id
AppId: MP-App1
Partner center APIs

Additional considerations

Conditional access

When it comes to managing your cloud resources, a key aspect of cloud security is identity and access. In a mobile-first, cloud-first world, users can access your organization's resources using a variety of devices and apps from anywhere. Simply focusing on who can access a resource is no longer enough. To master the balance between security and productivity, you also need to consider how a resource is accessed. By using Azure AD conditional access, you can address this requirement. With conditional access, you can implement automated access control decisions for accessing your cloud apps that are based on conditions.

See What is conditional access in Azure Active Directory? for more information.

IP range-based restrictions

You can restrict tokens to be issued to specific range of IP addresses only. This feature helps restrict the surface area of attack to a specific network only.

Multi-factor authentication

Enforcing multi-factor authentication helps restrict credential compromise situations by enforcing credentials verification to two or more forms. This feature allows Azure AD to validate the identity of the caller via secure secondary channels, such as mobile or email, before issuing tokens.

Please see: How it works: Azure Multi-Factor Authentication

References

Reference	URL
How to configure a new multi-tenant application	https://docs.microsoft.com/azure/active-directory/application-dev-setup-multi-tenant-app
Authorize access to Azure Active Directory web applications using the OAuth 2.0 code grant flow	https://docs.microsoft.com/azure/active-directory/develop/v1-protocols-oauth-code
How application consent works	https://docs.microsoft.com/azure/active-directory/application-dev-consent-framework
Microsoft Partner Center REST API reference	https://docs.microsoft.com/partner-center/develop/partner-center-rest-api-reference