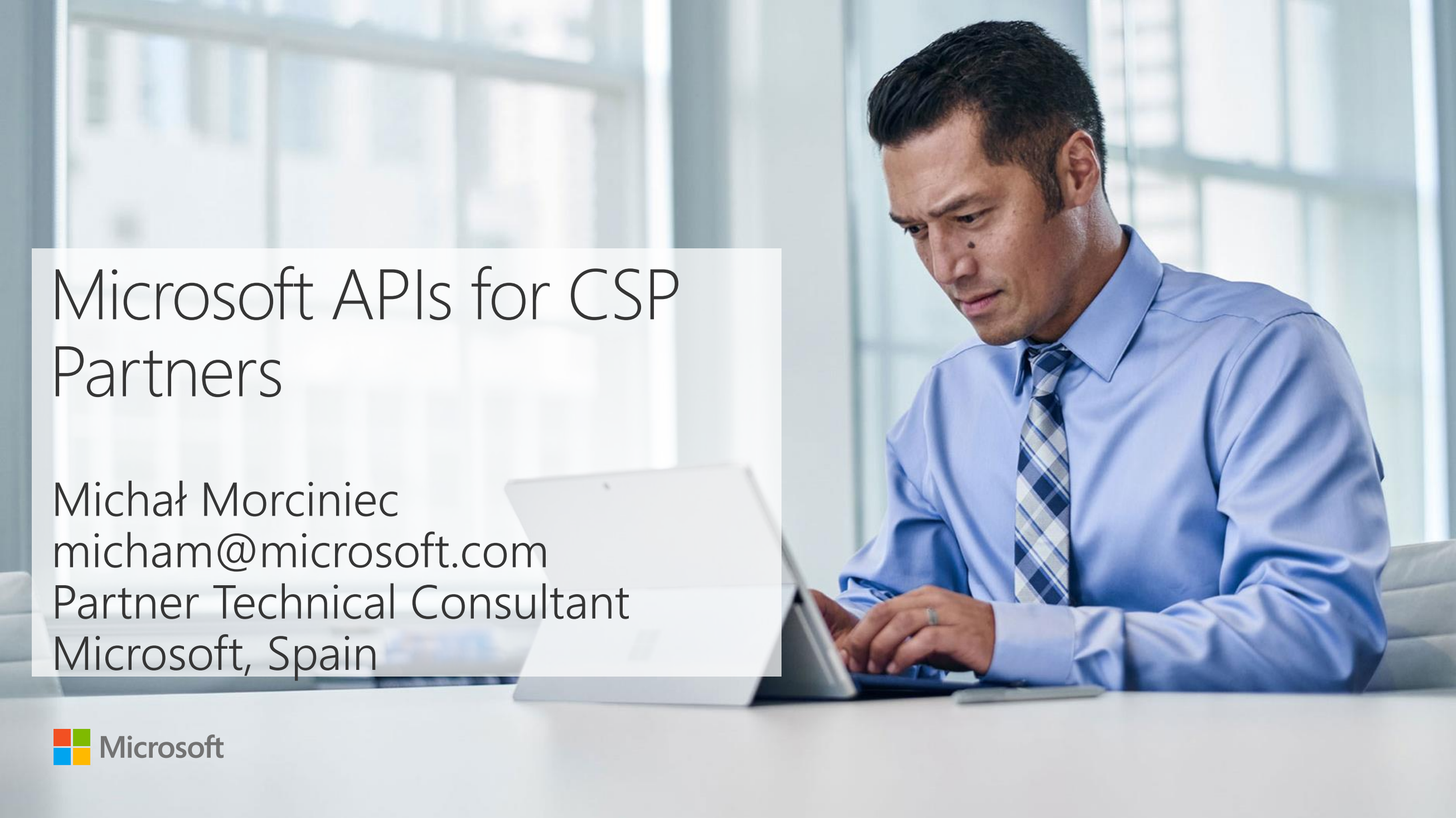


Microsoft APIs for CSP Partners

Michał Morciniec
micham@microsoft.com
Partner Technical Consultant
Microsoft, Spain

Microsoft

Agenda

- Integration Project Guidance
- Concepts Review
  - Common Microsoft APIs
  - Application Types in Azure AD
  - Application Permissions
  - Consent Framework
  - Pre-consent
  - Authentication types
- Partner Center API
- Azure AD Graph API
- Azure Resource Manager REST API

Objectives

- Scale to operate across customer Azure AD tenants with APIs (AD Graph, Azure Resource Management API)

- Explore more sample scenarios that may involve multiple API usage

Integration Project Guidance

Integration Guidance

Planning

1. Determine the type of integration required. This will scope the integration effort.
2. Determine the level of importance for each component being integrated to the business
3. Define business logic

Develop

1. Write code to provision customer
2. Write code to manage subscriptions
3. Ensure solution developed follows CSP Identity Guidance
4. Develop code for the integration defined in the planning stage

Test

1. Define internal testing team
2. Make any necessary modifications based upon feedback obtained from testing
3. Continue testing, making required changes, and obtaining feedback until an acceptable solution is compiled

Operate

1. Deploy the code to production
2. Track usage and consumption (e.g. AppInsights)
3. Continual Improvement

Concepts Review

Concepts to review

Common Microsoft APIs

Application Types in Azure AD

Permissions Scopes

Consent Framework

Pre-consent

Authentication Types: App Only or App+User

Common Microsoft APIs

Microsoft API Usage

Microsoft API	Service	Billing	Licensing	Provisioning	Support
Azure AD Graph API	Azure Office 365		✓ Assign licenses to users	✓ Contacts ✓ Domains ✓ Groups ✓ Users	
Office 365 Service Communications API	Office 365				✓ Office 365 service health
Microsoft Graph	Office 365		✓ Assign licenses to users	✓ Contacts ✓ Groups ✓ Users	
Partner Center API/SDK	Partner Center	✓ Azure rate card (CSP) ✓ Azure usage records ✓ CSP invoice details	✓ Assign licenses to users	✓ Assign users to directory roles ✓ Groups ✓ Indirect Model (2-tier) ✓ Subscriptions ✓ Tenants ✓ Users	✓ Service incidents (Azure/Office health) ✓ Service request management
Azure Insights REST API	Azure				✓ Azure service health
Azure Rate Card API	Azure	✓ Access Azure resource pricing (direct channel)			
Azure Resource Management API	Azure			✓ Azure service provisioning ✓ ARM tags	

Microsoft API Identification

API Name in Azure Mgt Portal *Documentation*	Microsoft API GUID (resourceAppID in Application Manifest)	Microsoft API REST URI Endpoint [resource]
Microsoft Partner Center *Partner Center API*	fa3d9a0c-3fb0-42cc-9193-47c7ecd2edbd	https://api.partnercenter.microsoft.com [https://api.partnercenter.microsoft.com]
Windows Azure Active Directory *Azure Active Directory Graph API*	00000002-0000-0000-c000-000000000000	https://graph.windows.net [https://graph.windows.net]
Microsoft Graph *Microsoft Graph*	00000003-0000-0000-c000-000000000000	https://graph.microsoft.com [https://graph.microsoft.com]
Windows Azure Service Management API *Azure Resource Management API*	797f4846-ba00-4fd7-ba43-dac1f8f63013	https://management.azure.com [https://management.core.windows.net]

Applications Types in Azure AD

Native Client Application

Description

Native applications in Azure AD are OAuth 2.0 public clients. A public client is an application that is not capable of keeping a secret secure. An example of this would be a desktop or mobile application since embedding passwords in those types of applications can be easily cracked. Since there is no secret for a native application it cannot obtain a token itself. App + User authentication must be utilized in order to obtain a token.

Usage

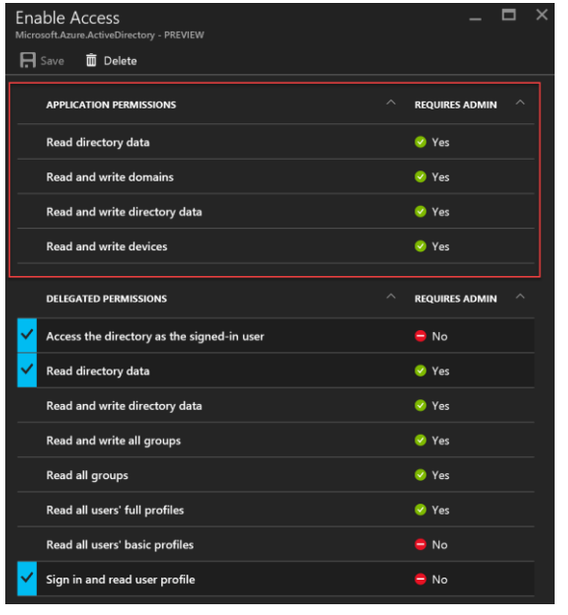
Native applications are primarily used by desktop and mobile applications where passwords cannot be easily secured. The following are common scenarios where native applications are leveraged:

- Daemons/Services
- Desktop application development
- Mobile application development

Comment

- Native applications are always multi-tenant
- Native applications have no secret (key)
- Native applications do not have App permissions - only Delegated permissions hence can only be used with App+User authentication

App+User



Web Application

Description

Web applications in Azure AD are OAuth 2.0 confidential clients. A confidential client is an application is capable of keeping a secret secure. This secret is used to identify the application during authentication scenarios.

Usage

Web applications are primarily used for web development where secrets can be easily secured. The following are common scenarios where web applications are leveraged

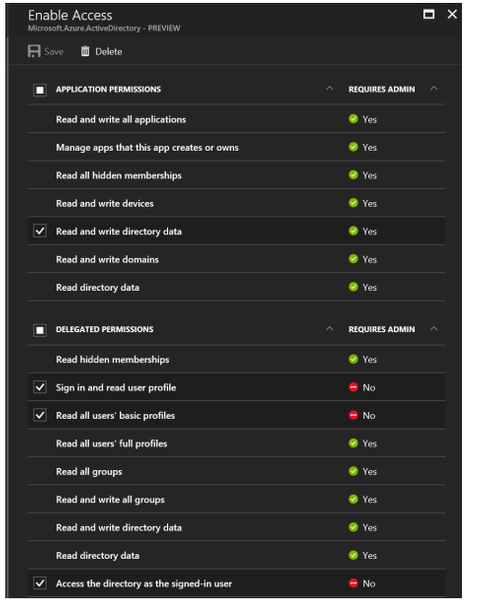
- Web API projects
- Websites

Comments

- Web app can be single or multi-tenant
- Web app has a secret (key, cert)
- It is capable Web app is capable of App only authentication.
- Both App Permissions and Delegated Permissions can be setup in API access if API supports it

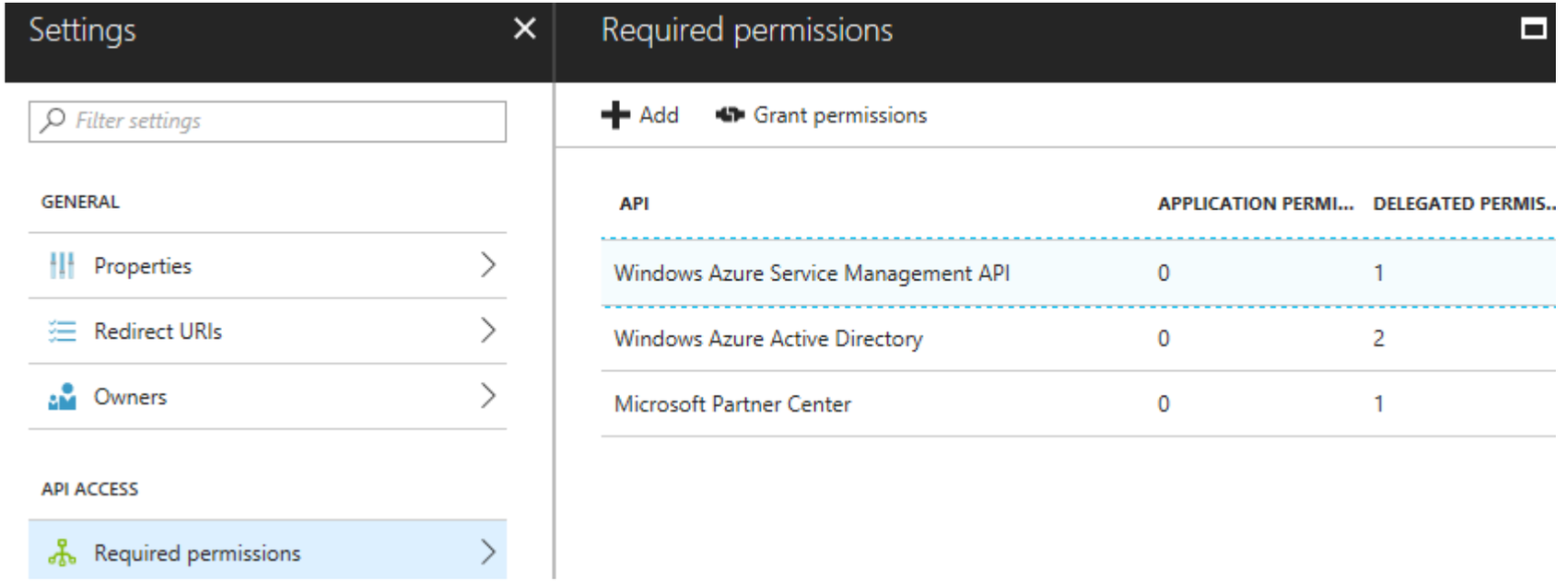
App Only

App+User



Application Permissions

Adding API Access for Application



API Permission Scopes

APIs expose OAuth2.0 permissions scopes

Some permissions require Administrator consent

Permissions are App-only (role) or delegated (scope)

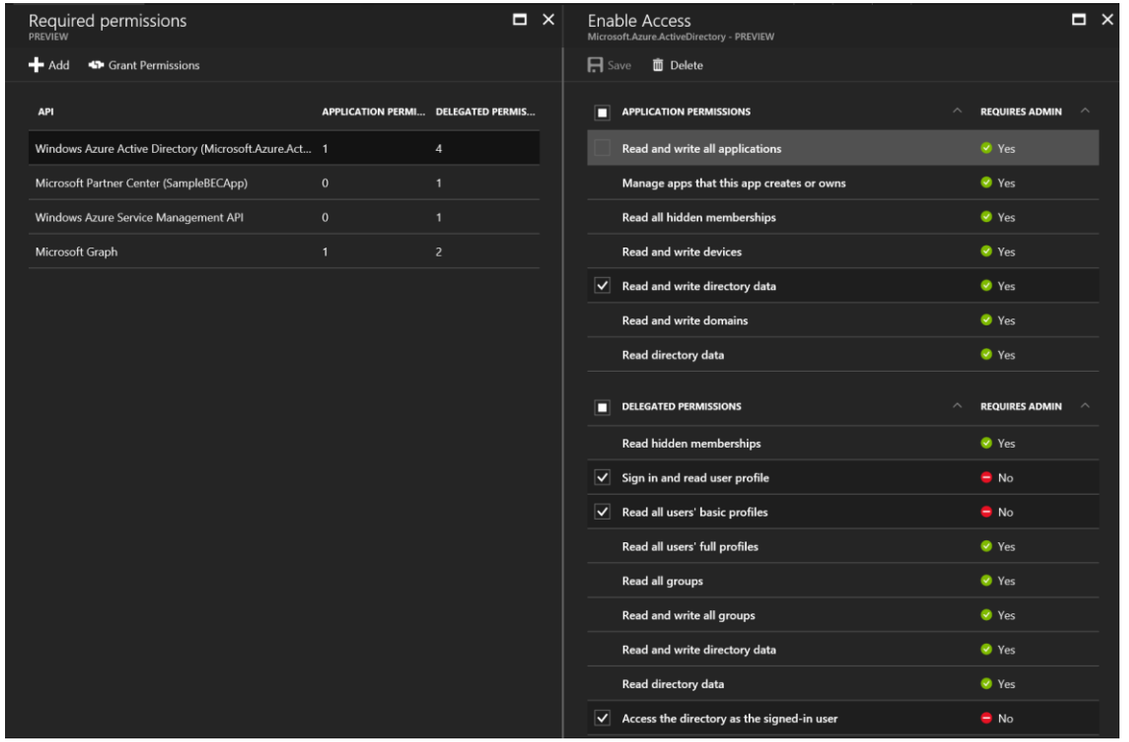
Permission description in the Azure portal	Identifier	Scope value	Type
Sign in and read user profile	311a71cc-e848-46a1-bdf8-97ff7156d8e6	UserProfile.Read	User
Read directory data	5778995a-e1bf-45b8-affa-663a9f3f4d04	Directory.Read	Admin (except for users from the tenant where the Application is defined)
Read and write directory data	78c8a3c8-a07e-4b9e-af1b-b5ccab50a175	Directory.Write	Admin
Access the directory as the signed-in user	a42657d6-7f20-40e3-b6f0-cee03008a62a	user_impersonation	Admin (except native clients)

https://msdn.microsoft.com/en-us/Library/Azure/Ad/Graph/howto/azure-ad-graph-api-permission-scopes

Delegated permissions require user sign-in

Effective permissions **is the intersection** of permissions granted by scope and the ones of user.

App-only scope grants the application permissions as configured

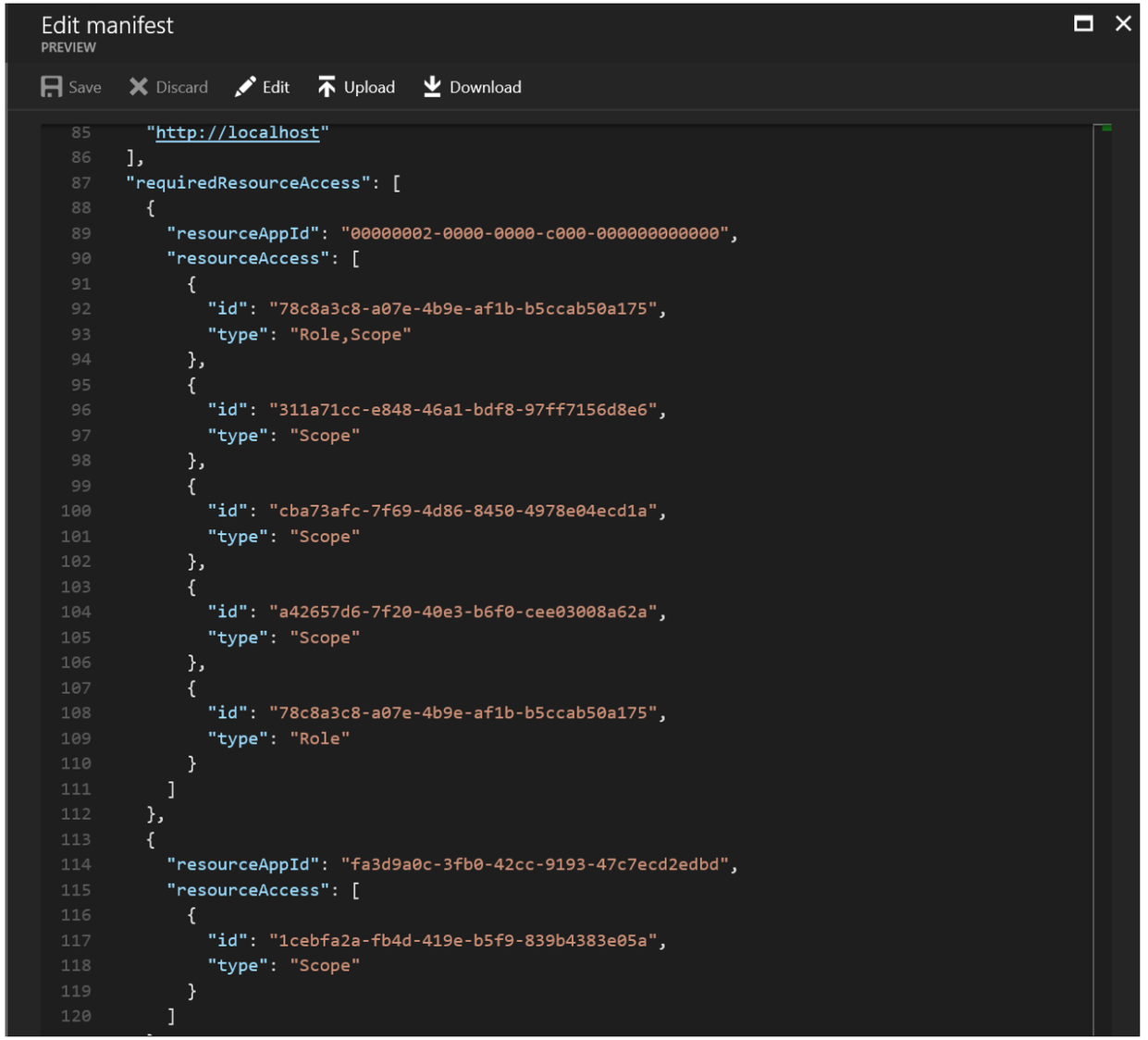


Application Manifest and API Permissions

Application manifests lists permissions to APIs that the Application will require.

Permission description in the Azure portal	Identifier	Scope value	Type
Sign in and read user profile	311a71cc-e848-46a1-bdf8-97ff7156d8e6	UserProfile.Read	User
Read directory data	5778995a-e1bf-45b8-affa-663a9f3f4d04	Directory.Read	Admin (except for users from the tenant where the Application is defined)
Read and write directory data	78c8a3c8-a07e-4b9e-af1b-b5ccab50a175	Directory.Write	Admin
Access the directory as the signed-in user	a42657d6-7f20-40e3-b6f0-cee03008a62a	user_impersonation	Admin (except native clients)

https://www.microsoftpressstore.com/articles/article.aspx?p=2473127&seqNum=2



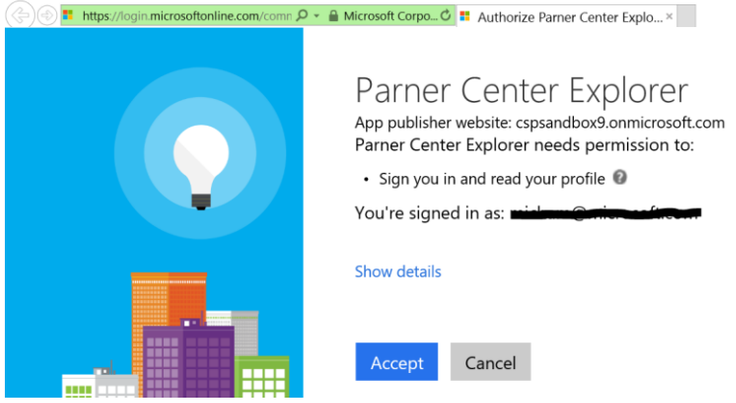
Demo

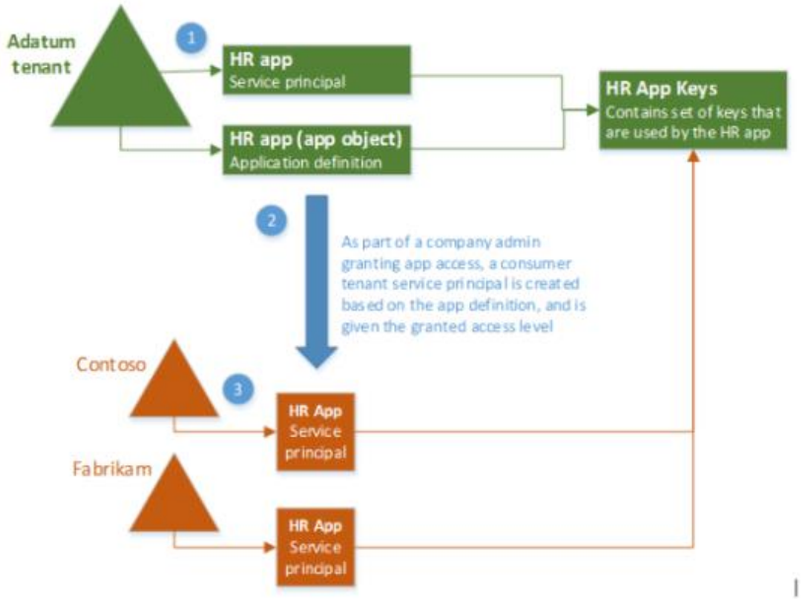
Application Registrations and Permissions
in the Partner Center

Consent Framework

Consent Framework

- Framework based on a user or administrator giving consent to an application
- Administrator can consent for all users
- Applies especially to multi-tenant applications
- Problematic in CSP partner scenarios
  - May have many customers and it is difficult to get Admin or individual users consent
  - Partner may need the application to provision and configure services in customer tenant before customer (Admin) is given access.





Pre-Consent

- Based on the fact the CSP partner already has delegated admin access to the customer tenant
- Once an application is configured for pre-consent the configuration will apply to all customers with which partner has reseller relationship (delegated administration).
- Other customers logging in will see consent dialogs
- Currently supported for applications requesting access to the following
  - Azure AD Graph API
  - Azure Resource Manager API
  - If you operate across Customer tenants with these APIs you will need to configure pre-consent

Pre-Consent script

```
Connect-AzureAD
$AppId = 'INSERT-APPLICATION-ID-HERE'
$DisplayName = 'INSERT-APPLICATION-DISPLAY-NAME-HERE'

$g = Get-AzureADGroup | ? {$_.DisplayName -eq 'AdminAgents'}
Get-AzureADGroupMember -ObjectId $g.ObjectId
$s = Get-AzureADServicePrincipal -All $true | ? {$_.AppId -eq
$AppId}

if ($s -eq $null) { $s = New-AzureADServicePrincipal -AppId $AppId -
DisplayName $DisplayName }
Add-AzureADGroupMember -ObjectId $g.ObjectId -RefObjectId
$s.ObjectId
```

https://github.com/Microsoft/Partner-Center-Explorer/blob/master/docs/Preconsent.md

Demo

Configuring Pre-consent for application

Authentication Types

App Only Authentication

Description

App Only authentication is where the client identifier and client secret are used to obtain a token from Azure AD for the application itself. This is possible because the application in Azure AD has a secret, also known as a password, that is used to verify the application's identity.

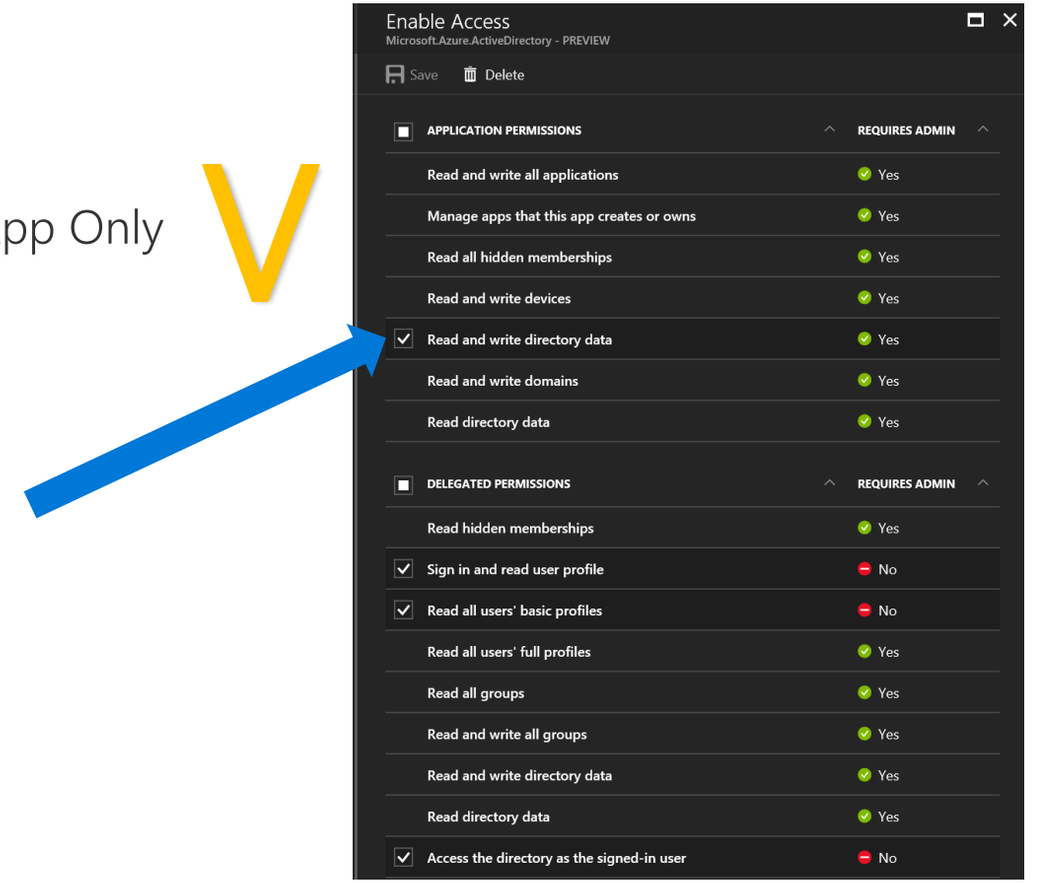
Usage

This type of authentication is typically used by APIs and websites. Since the application will be retrieving a token for itself permissions must be granted

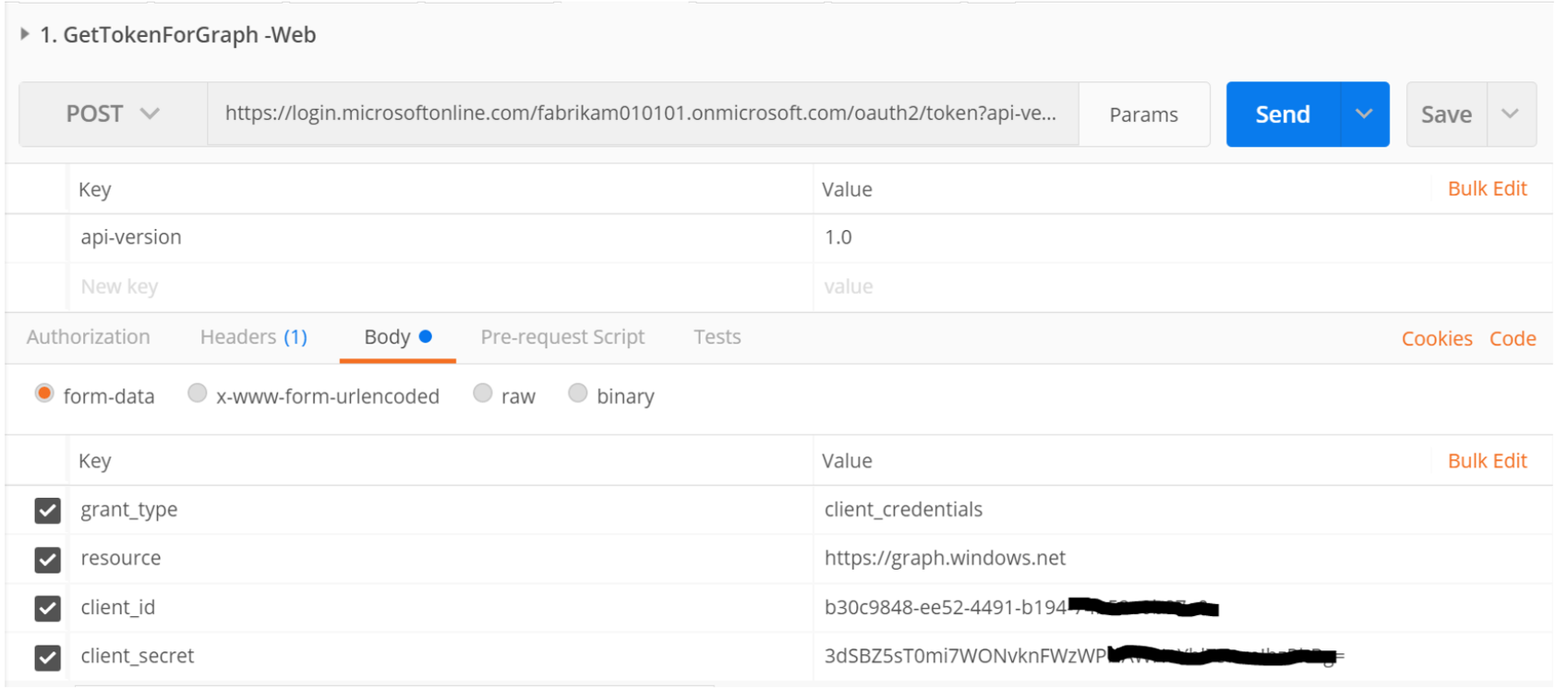
at the application level.

The figure to right highlights the permission set that corresponds to the App Only authentication type.

App Only



App-only authentication (Graph)



Client credentials grant flow

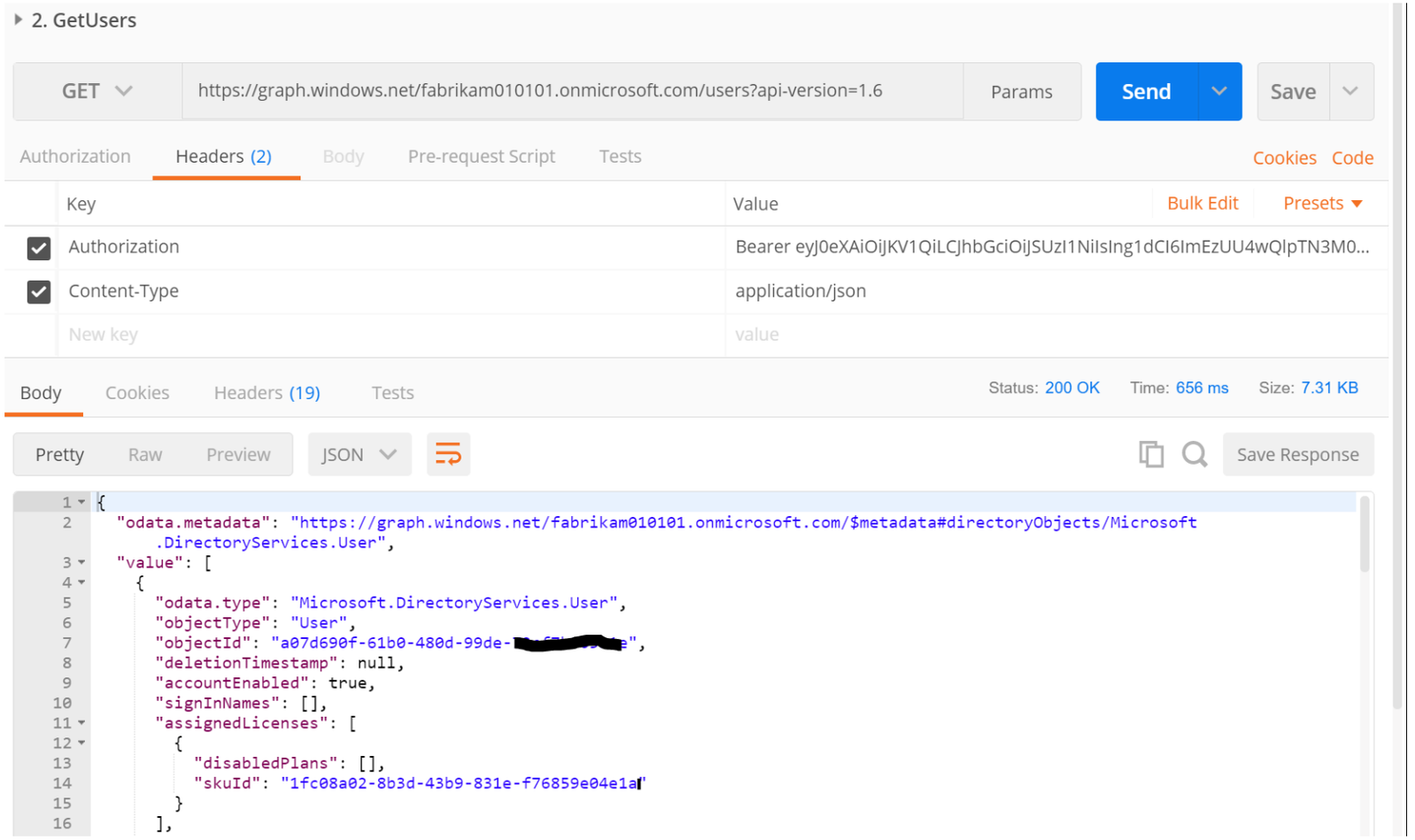
https://github.com/Microsoft/azure-docs/blob/master/articles/active-directory/develop/active-directory-protocols-oauth-service-to-service.md

Note

- The token endpoint URL identifies Customer tenant (fabrikam010101)
- we use web application so we need to provide both client_id (AppId) and client_secret (key)
- the Service Principal object representing application has to exist in directory (fabrikam010101)

(for multi-tenant apps it means that either the tenant Admin consented explicitly or pre-consent has been configured for Partner managed applications).

Perform Operation on Graph



Note the Access token is passed to the Authorization header

App + User Authentication

Description

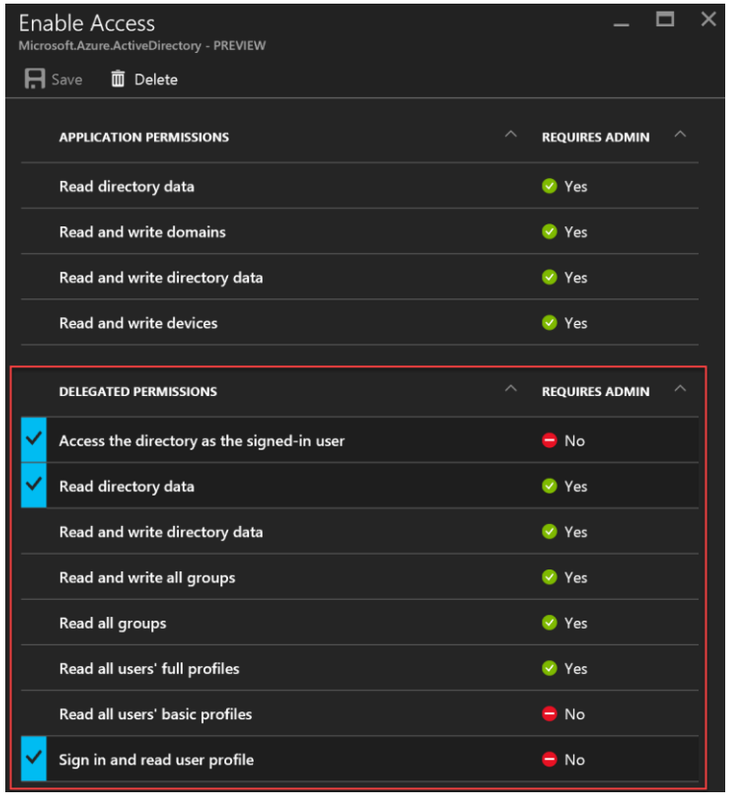
App + User authentication is where the client identifier and user credentials are used to obtain a token from Azure AD. This type of authentication requires user credentials because the application does not have a secret that can be used to verify the identity of the application.

Usage

This type of authentication is typically used in desktop and mobile applications (native) where secrets cannot be secured. Also, it is required for specific operations such deleting tenants and users.

The figure to the right highlights the Delegated permissions that correspond to the App + User authentication type.

App+User



App+User Authentication (ARM)

▸ 0. User+App Authentication ARM

POST https://login.microsoftonline.com/{{PC_CUSTOMER_DOMAIN}}/oauth2/token

Authorization Headers (1) Body ● Pre-request Script Tests ●

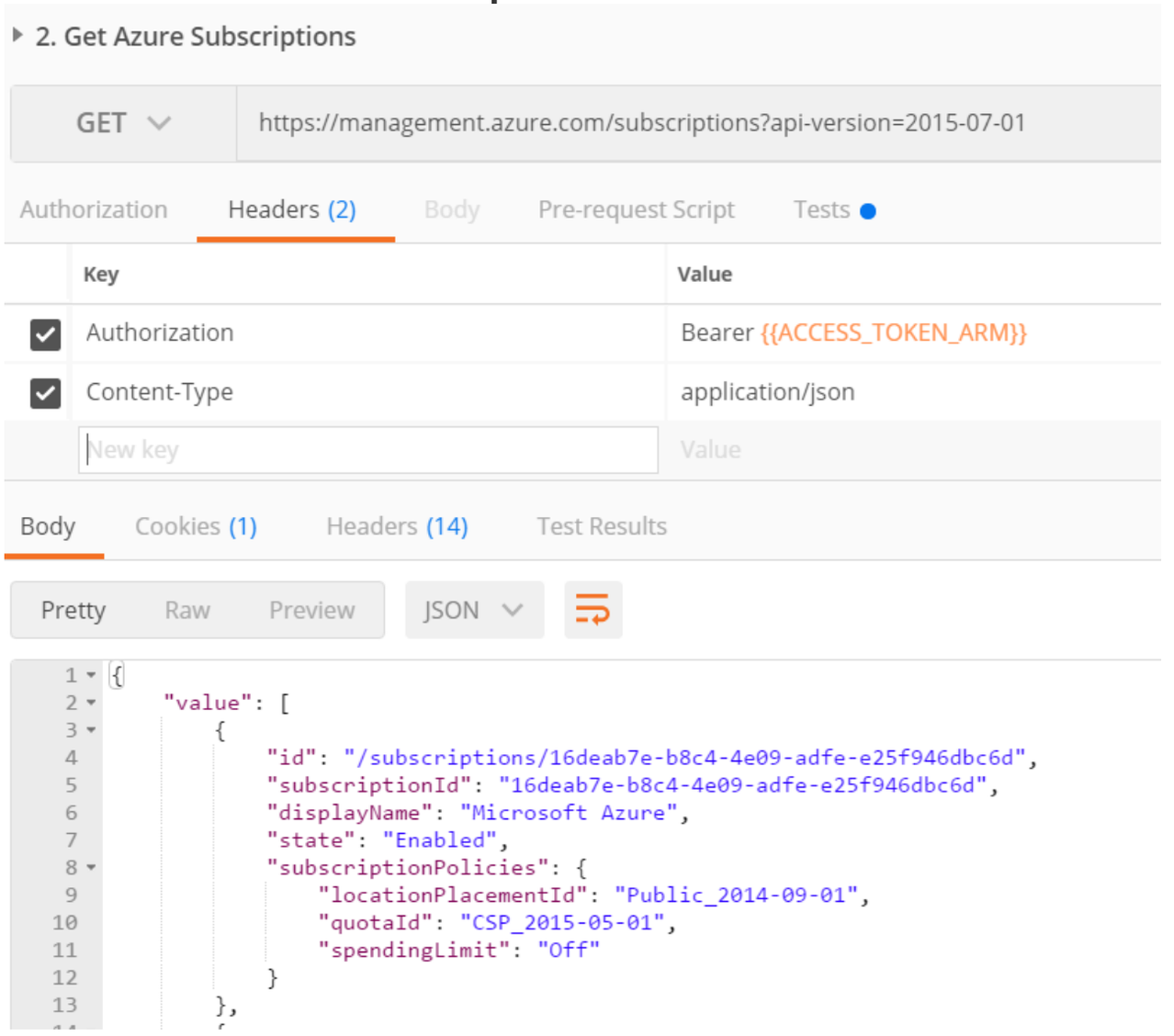
form-data x-www-form-urlencoded raw binary

Key	Value
resource	https://management.core.windows.net/
grant_type	password
username	{{PC_USER}}
password	{{PC_USER_PWD}}
scope	openid
client_id	{{PC_CLIENT_ID}}

- The login endpoint URL identifies Customer tenant (e.g.: fabrikam010101.onmicrosoft.com)
- Native application is used so we provide only client_id (AppId)
- the Service Principal object representing application has to exist in directory (fabrikam010101)

(either the tenant Admin consented explicitly or pre-consent has been configured for Partner managed application).

Perform Operation with ARM



Token is passed in the Authorization header

General Pattern in C# to obtain tokens with Partner Center credentials (App+User) for APIs

```
/// <param name="tenantId">the GUID or the domain of customer tenant
/// e.g.: “contoso.onmicrosoft.com” </param>
/// <param name="resource">the URI of the resource (API) to get access token for
/// e.g.: “https://management.core.windows.net/” </param>
/// <returns> access token to resource</returns>

private static async Task<AuthenticationResult> LoginToAad(string tenantId, string resource)
        {
var addAuthority = new UriBuilder(“https://login.microsoftonline.com/”+ tenantId);
            UserCredential userCredentials = new UserCredential(SettingsHelper.UserId,
                                                    SettingsHelper.UserPassword);
            AuthenticationContext authContext = new
AuthenticationContext(addAuthority.Uri.AbsoluteUri);
            return await authContext.AcquireTokenAsync(resource,
                                                       SettingsHelper.ClientId,
                                                       userCredentials);

        }
```

Partner Center API

Partner Center API

- .NET SDK and library for Visual Studio
  - Managed authentication tokens
  - Simple interface library for network calls with retries
- Sample code
  - Sample code for API calls
  - Sample sales application

- API functions
  - Directly list customers
  - List available offers
  - Raise support tickets in code
  - Rate usage in one API call
  - Validate tenant name availability
  - And more…

Partner Center Integration Sandbox

Purpose

A test environment used to test code in Partner Center so that you do not accidentally incur new charges that your company is responsible for paying.

Limitations

Invoices, Billing, Recon. Files, Azure Usage Records not available

Cannot request reseller relationship

Cannot create offer that require min 25 licenses

Limits

up to 75 Customers

5 subscriptions per Customer

25 licenses per subscription

$200 Azure usage per month

2 VM Reserved instances (B series) per Customer

Partner Center API Authentication

App

A token is obtained using either client credentials. When using App only authentication permissions must be assigned to the application itself in Azure AD using the *Application* permission scope.

App + User

A token is obtained using either user credentials or user assertion. When using App + User for authentication permissions must be delegated to the application itself in Azure using the *Delegate* permission configuration.

Some operations require the use of App + User authentication:

- Get Invoice by ID
- Create a service request (PC support ticket)
- Assign licenses to a user
- Reset user password for a customer
- Delete a user account for customer

Partner Center API Demo

Get usage records of Azure subscription via APIs
Given CustomerId and Azure subscription Id

- Partner Center API to enumerate usage records
- Extract resource tags from usage record
- Use ARM API to check actual resource tags

Azure AD Graph API

Azure AD Graph API
Overview

- RESTful interface for Azure Active Directory
  - Programmatic access to directory objects such as contacts, groups, roles, tenant information, and users
  - Tenant specific – queries are scoped to the individual tenant context
- Requests use standard HTTP methods
  - Compatible with OData V3
  - DELETE, GET, PATCH, POST to created, delete, read, and update
  - Response support JSON, XML, stand HTTP status codes

Azure AD Graph API

Scenarios

- Reading from Azure Active Directory
  - Check subscriptions
  - List of customers
  - Read directory metadata
- Writing to Azure Active Directory
  - Add users to groups or roles
  - Assign licenses
  - Provision or deprovision users
  - Update groups and users
- “Destructive transactions” require App + User auth

Navigating the REST API

```
https://graph.windows.net
      /{tenant}/{entity-set}/{id}/{property}?api-version={version}

Get a tenant-level entity
      /user
Select a member from the entity set
      /user/{id}
Get an entity property
      /groups/{id}/displayName
Traverse to related entity type via navigation properties
      /groups/{id}/members
```

Azure AD Graph API

Authentication

- Active Directory Authentication Library
  - Library used to authenticate applications and users
  - Extensible token cache management

- Can be used for single sign-on for
  - Azure
  - Office 365
  - Third party applications that utilize Azure AD for authentication

https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-authentication-libraries

Demo

Add admin user as owner of Azure subscription via APIs
Given CustomerId and Azure subscription Id

- Use Azure AD Graph API to get object Id of admin user
- Use ARM API to get Id of owner role
- Use ARM API to assign the user as owner for the subscription

Azure Resource Manager REST API

Azure Resource Manager REST API

Overview

- RESTful interface for Azure Resource Manager
  - Programmatic ability to manage resources through Azure Resource Manager
  - Tenant specific – operations are scoped to the individual tenant context
- Requests use standard HTTP methods
  - Each operation returns a x-ms-request-id header that can be used for troubleshooting
  - Supports cross-origin resource sharing (CROS)

Azure Resource Manager
Authentication

- Azure Active Directory is the identity provider
- Authorization tokes can be obtained two different ways
  - App-only: In this case the Azure AD application authenticates itself against Azure AD using the client identifier and client secret.
  - App + User: **This is the admin-on-behalf-of scenario**. The token obtained using this approach will be linked to the specified user.

Azure Resource Manager
Resource Groups

- Container that holds related resources
- Important factors to consider
  - All of the resources in a group should share the same lifecycle
  - Can be used to scope access control for administrative actions
  - Can contain resources that reside in different regions
  - Each resource can only exists in one resource group

Azure Resource Manager
Templates

- Templates are
  - JSON that describes resources, dependencies, and connections
  - Source files that can be checked into source control
- Templates can
  - Ensure uniformity of deployments
  - Provide cross-resource configuration and update support
  - Simplify orchestration
  - Simplify roll-back

Azure Resource Manager
Template Resources

- Authoring Azure Resource Manager templates: https://azure.microsoft.com/en-us/documentation/articles/resource-group-authoring-templates/
- Azure Resource Explorer: https://resources.azure.com
- Azure Quickstart Templates: https://azure.microsoft.com/en-us/documentation/templates/
- Azure DevTest Labs Templates: https://github.com/Azure/azure-devtestlab/tree/master/Samples
- Azure Resource Manager Template Visualizer (ArmViz): http://armviz.io/

Demo (REST)

Create storage account in Azure subscription using ARM
Given Customer Id and Azure subscription Id:

- Create Azure Resource Group in the subscription
- Validate ARM template deployment
- Deploy services via ARM template
- Query for deployment status

Resources

Azure AD Graph API	The Azure Active Directory Graph API provides programmatic access to Azure Active Directory through REST API endpoints. Apps can use the Azure AD Graph API to perform create, read, update, and delete (CRUD) operations on directory data and directory objects, such as users, groups, and organizational contacts. Additional information - https://msdn.microsoft.com/en-us/library/azure/ad/graph/api/api-catalog
Azure Rate Card API	The Azure Billing APIs provide access to resource consumption and metadata information for Azure subscriptions, providing the ability to better predict and manage Azure costs. Additional information - https://azure.microsoft.com/en-us/documentation/articles/billing-usage-rate-card-overview/
Azure REST APIs	Azure Resource Manager REST APIs are the heart of interacting with Azure, and form the connecting glue between your applications and Azure. They are described by specifications conforming to the OpenAPI Specification (formally known as Swagger 2.0). The OpenAPI specification provides a standard, language-agnostic interface to REST APIs. Additional information - https://azure.github.io/projects/apis/
Cloud Solution Provider Developers	This course is for Microsoft partners who are in the Cloud Solution Provider (CSP) program and who are interested in the Partner Center SDK and the Partner Center REST API. This includes partners who have been using the previous CREST API. The course consists of multiple video lessons and demos that are 10-60 minutes in duration. It also includes a self-paced hands-on-labs for course participants which is located at http://aka.ms/pcsdkhol. Additional information - https://channel9.msdn.com/Series/cspdev
Microsoft Graph	Microsoft Graph (previously called Office 365 unified API) exposes multiple APIs from Microsoft cloud services through a single REST API endpoint (*https://graph.microsoft.com*). Using the Microsoft Graph, you can turn formerly difficult or complex queries into simple navigations. Additional information - https://graph.microsoft.io/en-us/docs
Partner Center API Reference	The Partner Center managed API helps Cloud Solution Provider partners integrate their existing CRM or billing software with the Microsoft systems that manage customer accounts, place orders, manage subscriptions, and handle support requests in Partner Center. The Managed API also includes token management (so that you don't have to refresh your Azure AD tokens and authentication each hour) and a simple interface library for network calls with retries. Additional information - https://msdn.microsoft.com/en-us/library/partnercenter/mt635943.aspx
Partner Center Explorer	Sample project developed to demonstrate how to utilize the Azure AD Graph API, Azure Resource Management API, and Partner Center SDK to retrieve data for a given CSP partner and the respective customers. Additional information - https://github.com/Microsoft/Partner-Center-Explorer/

Next Steps

- Download webcast material from GitHub

- Register for Partner Center Early Adopter Program

- Open 1:1 API Consultation request if more help is needed
- (askptc@microsoft.com)

Contact Us

Technical services to help you win more deals, accelerate deployment and increase consumption

Learn more about your technical support benefits

http://aka.ms/MySupport

Microsoft Azure

http://aka.ms/PartnerTechnicalServicesAzure

Microsoft CRM Online

http://aka.ms/PartnerTechnicalServicesCRMOnline

Microsoft Office 365

http://aka.ms/PartnerTechnicalServicesO365

Microsoft Windows 10 and Enterprise Mobility Suite

http://aka.ms/PartnerTechnicalServicesWin10EMS

Microsoft