



# Spark the future.

## Microsoft Ignite

May 4 – 8, 2015  
Chicago, IL





# Office 365 Developer On Ramp (Part 2)

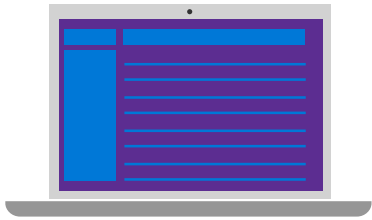
Andrew Connell  
MVP

[www.andrewconnell.com](http://www.andrewconnell.com)

[@andrewconnell](https://twitter.com/andrewconnell)

[github.com/andrewconnell/pres-o365-devrampup](https://github.com/andrewconnell/pres-o365-devrampup)

# Developer Program Launch



E-mail  
Newsletters



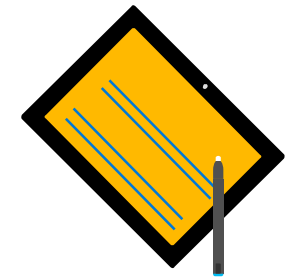
Free  
Developer  
Subscription



Free  
Training



Free  
Tools



Webinars

<http://developer.ondevprogram>

# Call to action



Sign up for  
Developer Program



Check out the  
express talks



Collect your  
stickers at Ignite

# Topics

- OAuth2 & OpenID Connect Concepts
- Azure Active Directory
- Office 365 APIs



# Oauth & Authentication Concepts

# What is OAuth2

- Authentication protocol
- Once user is authenticated, they obtain access tokens
  - Digitally signed string containing claims
- Access tokens are the “keys” to secured resources
- Secured resources trust the issuer of tokens

# Terminology

- Access token
- Refresh token
- OAuth flows
- Resource
- Issuer
- User / application



# It all Comes Down to Access Tokens

- For the app developer, the goal is to obtain an access token
  - Valid for a specific period of time
- Sometimes acquisition of an access token includes an refresh token
  - Refresh tokens can be used to update access tokens
- This token is passed with each HTTP request to authenticate the user and / or the app
  - Included in the HTTP request header:
  - `Authorization: Bearer [token]`

# Obtain Access Tokens with OAuth Flows

- Obtain access tokens using different defined "OAuth Flows"
  - Multiple flows defined in OAuth 2.0 spec
  - Both the Issuer & Resource must support the flow to use it
- Flow Options with Office 365 & Azure AD
  - Authorization Code Flow
  - Client Credentials Flow (App only)
  - Implicit Flow

# Flow #1 – Authorization Code

- Most common
- Very secure - application never gets user's creds
- Flow
  - Web app redirects user to Azure AD to login
  - Upon successful login, Azure AD redirects user back to web app with an authorization code
  - Web app uses this code to request access token on behalf of user
- Scenarios
  - Web applications that use federated logins
  - User interaction present

# Flow #2 – Client Credentials

- Very powerful
- Requires global tenant admin consent
  - Not user consent
- Flow
  - App configured with the public part of a certificate
  - App submits an encrypted request to the issuer requesting access token
  - Issuer returns access token back to requestor
- Scenarios
  - App only rights
  - Zero user interaction
  - Service / daemon processes

# Flow #3 – Implicit Flow

- Must be deliberately set...
  - Apps don't support it by default
- Enables OAuth2 authentication for apps that are 100% client-side
- Scenarios
  - Client-side application

# OpenID Connect – What is It?

- OAuth 2.0 is simply an authentication protocol that returns access tokens
  - No information about the user is included in the response
  - For instance no details like “who just logged in”
  - If you want identity details, you have to roll your own solution
- OpenID Connect adds identity to OAuth 2.0 process
  - Very lightweight wrapper ... unlike `ws-fed`
  - Id token returned by Azure AD with basic claims about the user

<http://openid.net/specs/openid-connect-core-1.0.html>



# Azure Active Directory

# Office 365 & Azure AD Directory

- Office 365 has a dependency on Azure AD
  - Every Office 365 tenant has its own Azure AD directory
  - Users in Office 365 tenants are stored in Azure AD directories
- Azure AD stores users for Office 365
- Also application configurations
- Office 365 trusts Azure AD
  - Azure AD applications can be granted permissions to SharePoint Online & Exchange Online data within Office 365



# Azure AD & Azure AD Applications

- Enables support for validating users & applications
  - Users = username & password
  - Application = client id & key
- Supports OAuth 2.0 & OpenID Connect
  - OAuth 2.0 - base protocol for authentication
  - OpenID Connect – thin layer around OAuth 2.0 that adds user identity
- Upon successful authentication, user / application obtains an OAuth 2.0 access token
  - Included in every request within HTTP header
  - Like cash... anyone can use it

# Properties of Azure AD Applications

- Name
- Sign-On URL
- Logo
- Single or Multi-Tenant
- Client ID
- Keys
- App ID URI
- Reply URL
- Permissions

# Creating Azure AD Applications

- Visual Studio 2013
  - Office Developer Tools for Visual Studio 2013
  - Connected Service wizard
- Azure Management Portal web interface



# Demo – Creating AzureAD Apps

# Single vs. Multi-Tenant Applications

## Single Tenant Apps

- Available to all users in your Azure AD directory
- Not available to users outside your Azure AD directory
- Typically internal apps for your organization's users

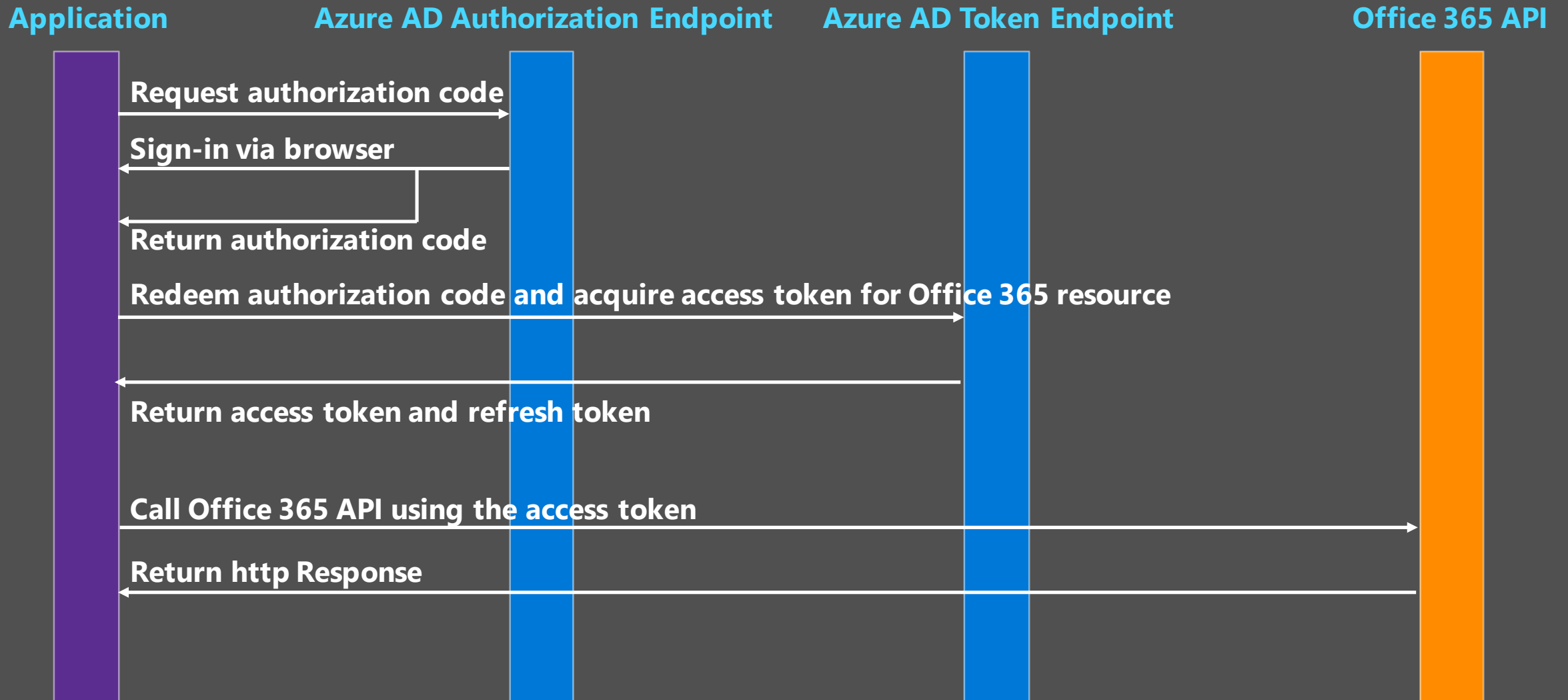
## Multi-Tenant Apps

- Just like single tenant apps except...
- Available to all users in any Azure AD directory
- Typically built by ISVs or as a SaaS offering



# OAuth2 & AzureAD Authentication Flows

# Authentication Flow Overview



# Azure AD OAuth Authorization Endpoint

`https://login.microsoftonline.com/<ad-tenant-id>  
/oauth2/authorize`

`https://login.microsoftonline.com/common  
/oauth2/authorize`

`?client_id=c5a5591c-bb7d-4b7c-944a-aaa17fa068aa  
&redirect_uri=https://www.foo.com/auth  
&response_type=code`



# Azure AD OAuth Token Endpoint (Access)

`https://login.microsoftonline.com/<tenant-id>  
/oauth2/token`

`https://login.microsoftonline.com/common  
/oauth2/token`

`?client_id=c5a5591c-bb7d-4b7c-944a-aaa17fa068aa  
&redirect_uri=https://www.foo.com/auth  
&client_secret=z19jGC3TmArM4aDg1C1GSVBsfoD5y5...  
&grant_type=authorization_code  
&code=AAABAAAAAvPM1KaPlrEqdFSBzjqfTGBzE2fzOMjs...  
&resource=https://graph.windows.net`

# Azure AD OAuth Token Endpoint (Refresh)


`https://login.microsoftonline.com/<tenant-id>  
/oauth2/token`

`https://login.microsoftonline.com/common  
/oauth2/token`

`?client_id=c5a5591c-bb7d-4b7c-944a-aaa17fa068aa  
&redirect_uri=https://www.foo.com/auth  
&client_secret=z19jGC3TmArM4aDg1C1GSVBsfoD5y5...  
&grant_type=refresh_token  
&code=AAABAAAAvPM1KaPlrEqdFSBzjqfTGPI91jyYrww...  
&resource=https://outlook.office365.com`



# DEMO - Inspecting the Authentication Process Flow



# OpenID Connect

# OpenID Connect

`https://login.microsoftonline.com/<tenant-id>  
/oauth2/authorize`

`https://login.microsoftonline.com/common  
/oauth2/authorize`

`?client_id=c5a5591c-bb7d-4b7c-944a-aaa17fa068aa  
&redirect_uri=https://www.foo.com/auth  
&response_type=code+id_token`

# OpenID Connect's ID Token

```
{  
  aud: "c0d204d8-1f5b-424e-944c-489cab675586",  
  iss: "https://sts.windows.net/4254573b-5de2-452e-b706-30eed7ce1ebf/",  
  nbf: 1421785651,   exp: 1421789551,  
  upn: "andrew.connell@acio36503.onmicrosoft.com",  
  sub: "1rxMpLjHWTQxMrsQDw8KtkwxICCft1L1UX29CFbf0TM",  
  given_name: "Andrew",   family_name: "Connell",  
  name: "Andrew Connell",  
  unique_name: "andrew.connell@acio36503.onmicrosoft.com",  
  pwd_exp: "7433009",  
  pwd_url: https://portal.microsoftonline.com/ChangePassword.aspx  
}
```



# App-Only Permissions

# Application & Delegated Permissions

## Delegated Permissions | User + App

- Apps & users have permissions
- Act on behalf of a user
- User (or tenant admin) grants app rights to app on it's behalf
- App is thus *delegated* permissions by the user
- App & user must both have permissions

## Application Permissions | App-Only

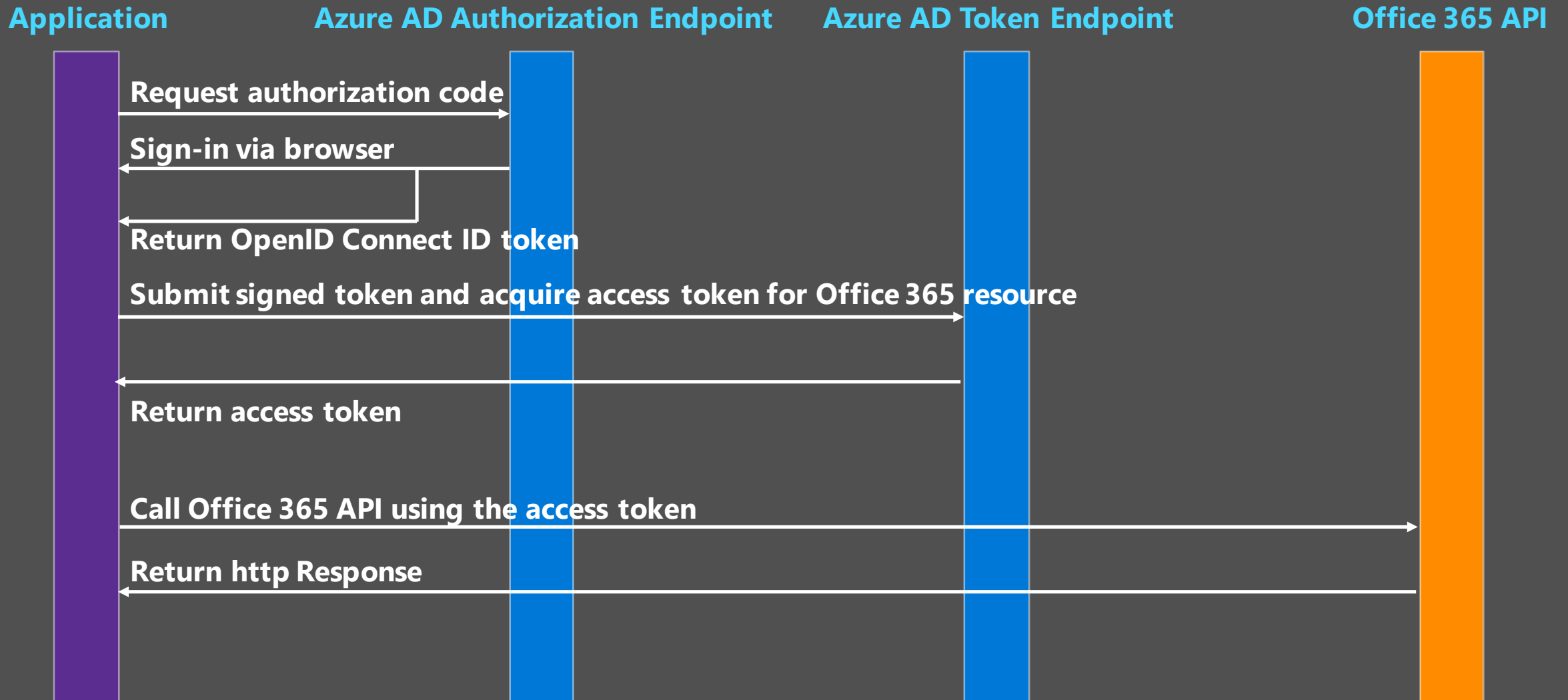
- Apps have permissions
- Acts independent of user
- Tenant admin grants app rights to all users within the organization
- Users do not need to grant app permissions
- App then acts independently of the users in the organization



# App-Only Characteristics

- Apps that utilize the app-only permission model have different configuration & usage requirements than those that leverage app+user permissions
- **App+User permission model:**
  - Apps identified by client ID & keys
  - Users grant app permissions & obtain access tokens after they authenticate
  - Use refresh tokens to acquire new access token without re-authenticating
- **App-Only permission model:**
  - Apps identified by client ID & certificate
  - Tenant admin grants app permission to all users in the organization
  - App obtains access tokens; refresh tokens not used

# App-Only Authentication Flow Overview



# App-Only – Authorization Endpoint Request


`https://login.microsoftonline.com/common  
/oauth2/authorize`

`?state=001adebf-3565-4745-b07f-d421f6ba84c2  
&response_type=code+id_token  
&scope=openid  
&nonce=e9615b42-3aff-44e4-a6cd-7bd7904f920a  
&client_id=65923aa3-ec8b-4f6c-acbb-ed2f90ecf59d  
&redirect_uri=https://www.foo.com/auth  
&resource=https://graph.windows.net/  
&prompt=admin_consent  
&response_mode=form_post`

# App-Only – Token Endpoint Request

`https://login.microsoftonline.com/<tenant-id>  
/oauth2/token`

`resource=https://graph.windows.net/  
&client_id=65923aa3-ec8b-4f6c-acbb-ed2f90ecf59d  
&client_assertion_type=  
urn:ietf:params:oauth:client-assertion-  
type:jwt-bearer  
&client_assertion=<signed jwt token>  
&grant_type=client_credentials`



# DEMO – App Only Permissions



# Office 365 APIs

# Office 365 APIs

- First released as version 1.0 late 2014
  - Sites API (SharePoint)
  - Contacts API
  - Calendar API
  - Mail API
  - Files API (OneDrive for Business)
  - Discovery Service
- All Office 365 APIs share some common characteristics
  - Azure Active Directory apps
  - Authentication & authorization
  - Versioning
  - SDK & OData / REST
  - Tooling

# Upon the v1.0 Release to Today...

- Each API had it's own endpoint
- Each API had it's own resource ID
- Needed separate access token for each endpoint



# Challenges Without the Discovery Service

- Critical parts of all Office 365 APIs:
  - Service resource ID
  - Service endpoint URL
- Some resource IDs & service endpoint URLs are predictable
  - Contacts, calendar, mail...
- Some are not predictable...
  - Files (aka: OneDrive for Business)

# How Does the Discovery Service Help?

- Returns a list of all Office 365 API services the user has access to
  - Services the user does not have access to are omitted
  - Omitted services – those the Azure AD application does not have rights to
- Includes the necessary information to connect to the services
  - Service resource ID
  - Service endpoint URL
- Discovery service well-known details:
  - Service resource ID = <https://api.office.com/discovery>
  - Service endpoint URL = <https://api.office.com/discovery/v1.0/me>



# DEMO – Discovery Service

# Office 365 Unified API

- Single API “proxy” to all underlying apps
  - Will remove the need for the Discovery Service
  - Currently in preview
- 
- Breakout Session “Office 365 Unified API (preview)”
    - Video from Build - [channel9.msdn.com/Events/Build/2015/3-641](https://channel9.msdn.com/Events/Build/2015/3-641)
    - Session @ Ignite: BRK3199, Tuesday @ 10:45 by Yina Arenas



# DEMO – Unified API

# Exchange Online Related APIs

Contacts  
API

Calendar  
API

Mail API

# Check the Office Dev Training Content

- Most of the following samples are pulled from Office Dev's training content
  - Hands on labs
  - Links to videos
  - Completed lab samples
- [dev.office.com/training](https://dev.office.com/training)
- [github.com/OfficeDev/TrainingContent](https://github.com/OfficeDev/TrainingContent)



# DEMO – Exchange Online APIs



# Sites API & Files API

## ■ Sites API

- Exposes parts of the SharePoint Online REST API
- Data within lists

## ■ Files API

- Data within libraries
- OneDrive for Business



# DEMO – Sites API & Files API

# Additional APIs Recently Announced

- Office Graph API
  - Interact with the API that powers Delve
- Video Portal API
  - View & upload videos to channels
  - View contents of channels
- Groups API
- OneNote API

# Summary

- Enterprise JavaScript
  - Enterprise Services Architecture
  - Enterprise Frameworks
  - OAuth2 Primer
  - Azure Active Directory
  - Office 365 APIs
- 
- [github.com/andrewconnell/pres-o365-devrampup](https://github.com/andrewconnell/pres-o365-devrampup)

