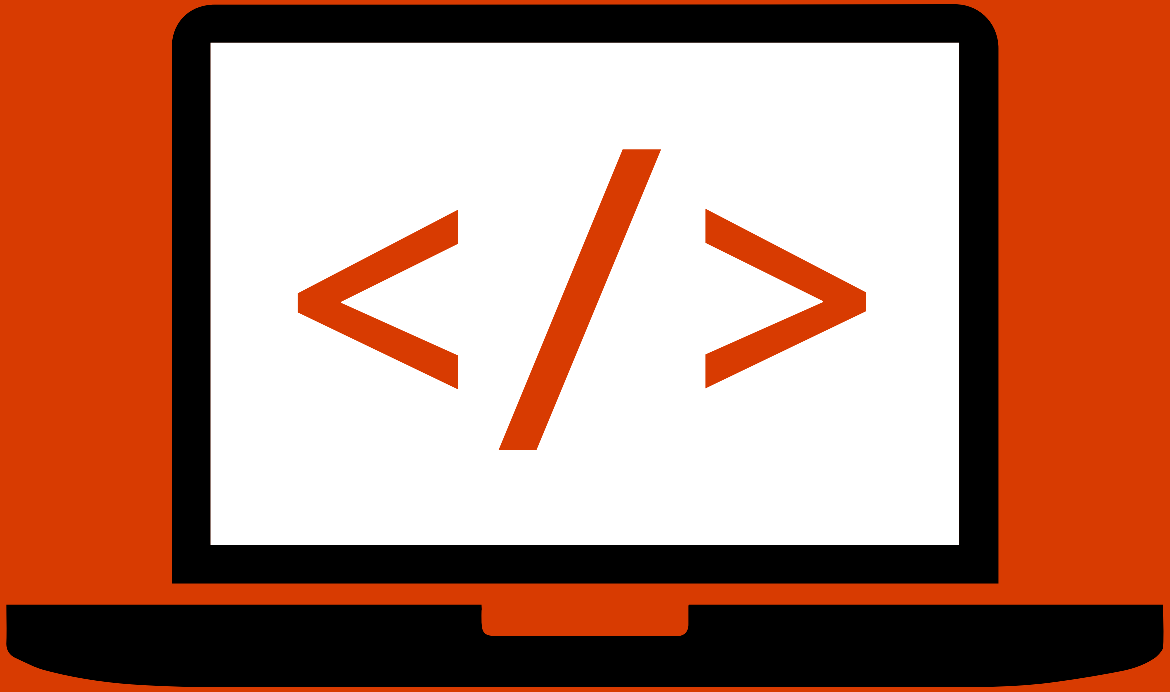
Getting Started in Office 365 with the Microsoft Graph API

Tiago Costa
MVP – Office Server and Services
MCT and MCT Regional Lead

Office

http://dev.office.com/

Intro to the Microsoft Graph API



Agenda

- Intro to the Microsoft Graph API
- Getting started
- SDKs and Code Samples
- Resources

1

Intro to the Microsoft Graph API

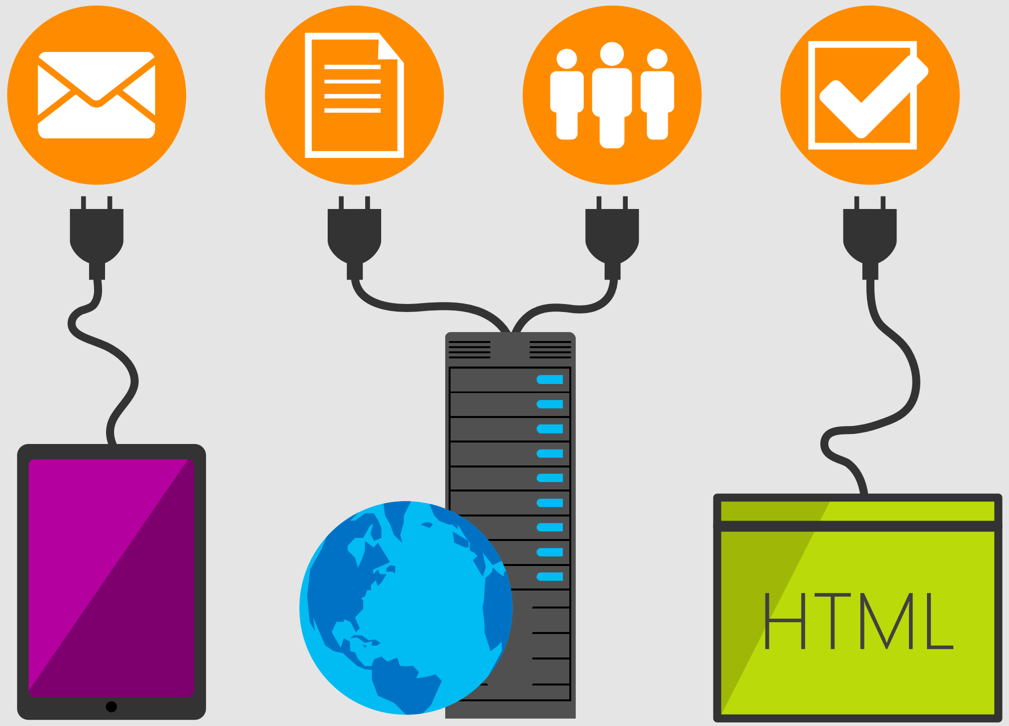
Office

http://dev.office.com/

Developer vision

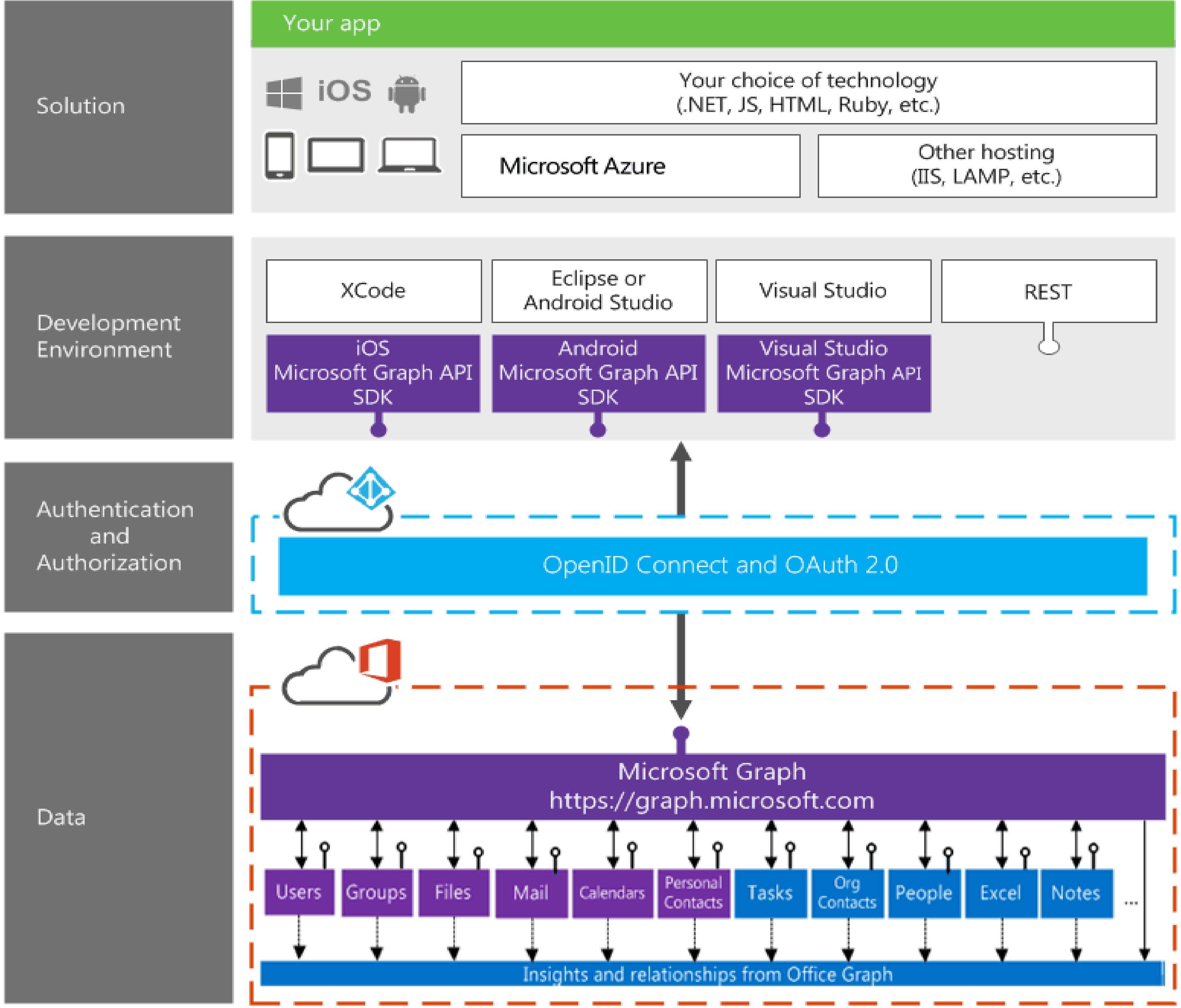
USERS

DATA



Office

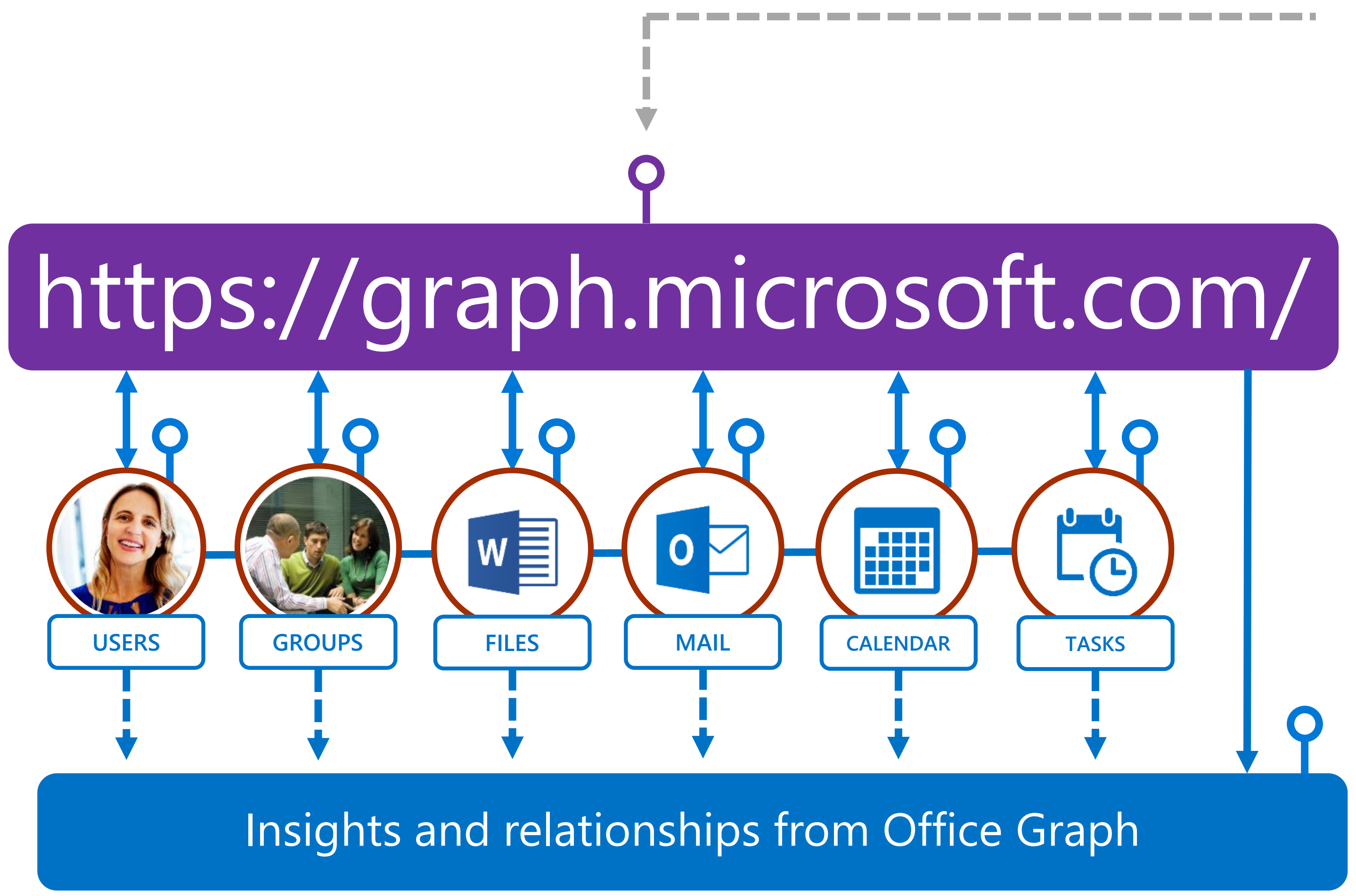
Building integration with Office 365



Solution
Your app
iOS
Your choice of technology (.NET, JS, HTML, Ruby, etc.)
Microsoft Azure
Other hosting (IIS, LAMP, etc.)
Development Environment
XCode
Eclipse or Android Studio
Visual Studio
REST
iOS Microsoft Graph API SDK
Android Microsoft Graph API SDK
Visual Studio Microsoft Graph API SDK
Authentication and Authorization
OpenID Connect and OAuth 2.0
Data
Microsoft Graph https://graph.microsoft.com
Users
Groups
Files
Mail
Calendars
Personal Contacts
Tasks
Org Contacts
People
Excel
Notes
...
Insights and relationships from Office Graph

Office 365

Microsoft Graph API


https://graph.microsoft.com/
USERS
GROUPS
FILES
MAIL
CALENDAR
TASKS
Insights and relationships from Office Graph

Office

Microsoft Graph, gateway to Office 365

Single resource that proxies multiple Microsoft services

Allows for easy traversal of objects and relationships

Simplifies token acquisition and management

Eliminates the need to traditional discovery (using “me” and “myorganization”)

Office

Office 365 direct API Endpoints

Direct API endpoints for all the Office 365 Services may also be invoked

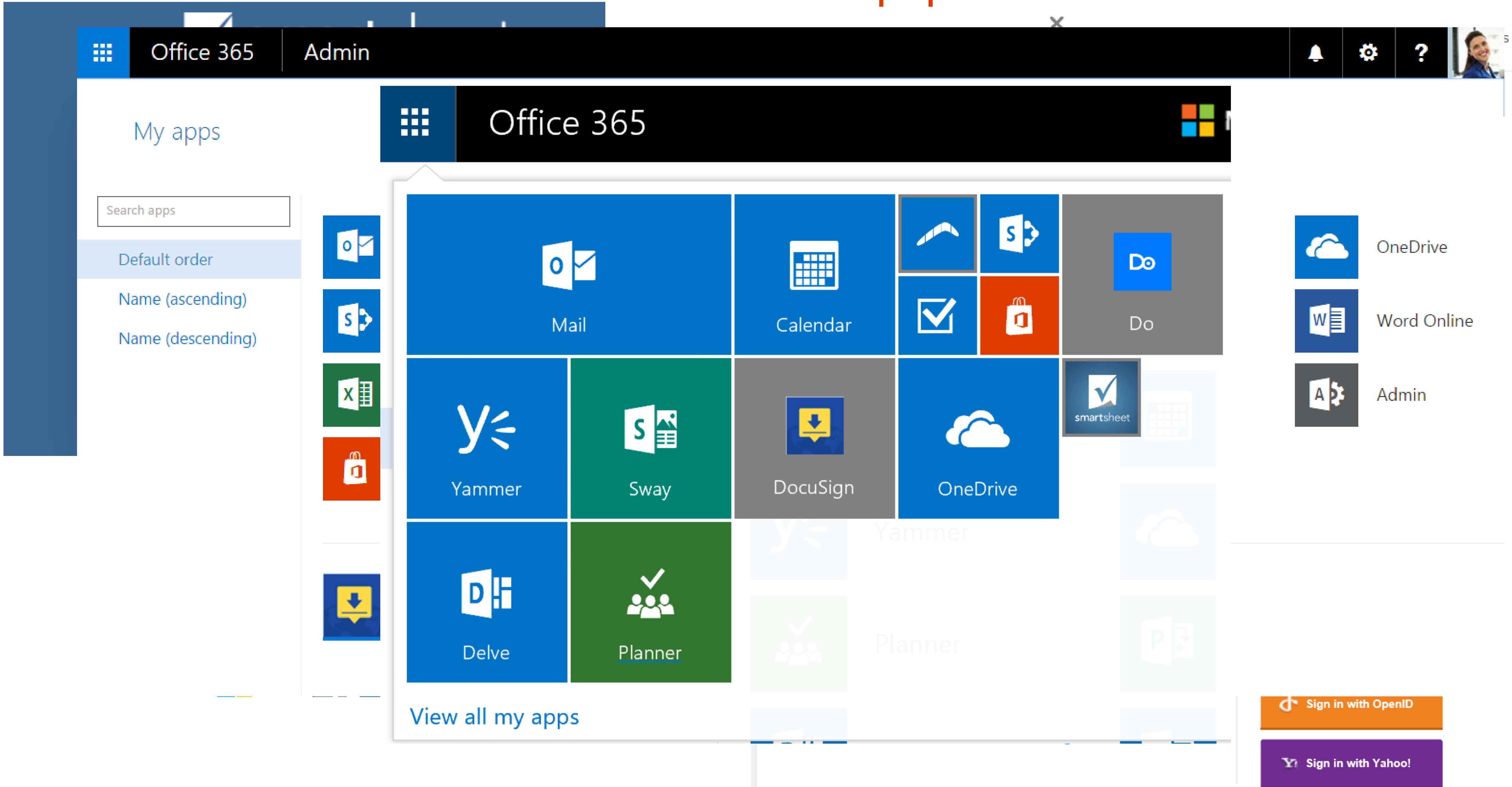
- Outlook, OneDrive, OneNote, etc.

Direct endpoints have new functionality before it is exposed via the Graph API

- Examples:
  - Outlook web hooks
  - Time zone on calendar

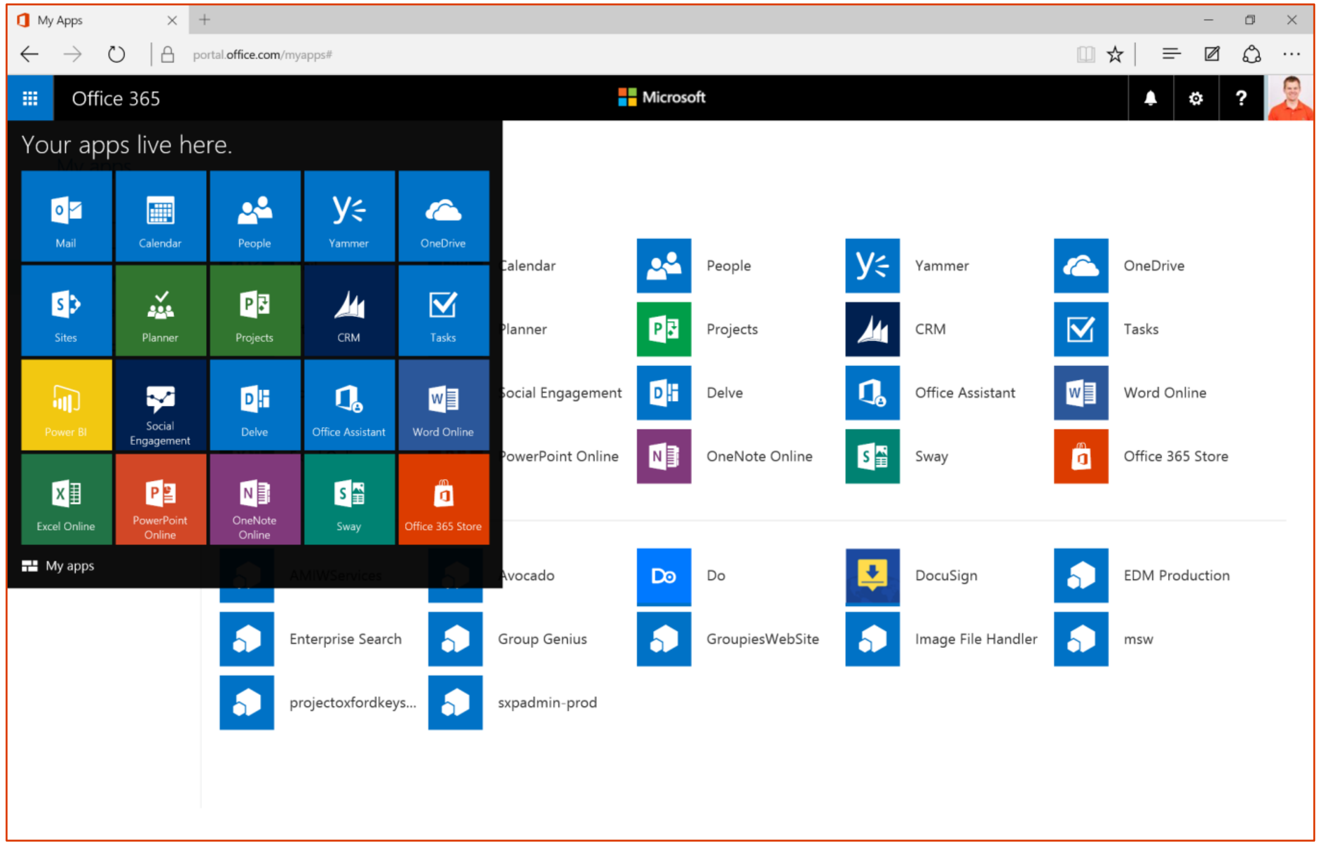
Office

Office 365 connected apps



Single authentication flow for Office 365

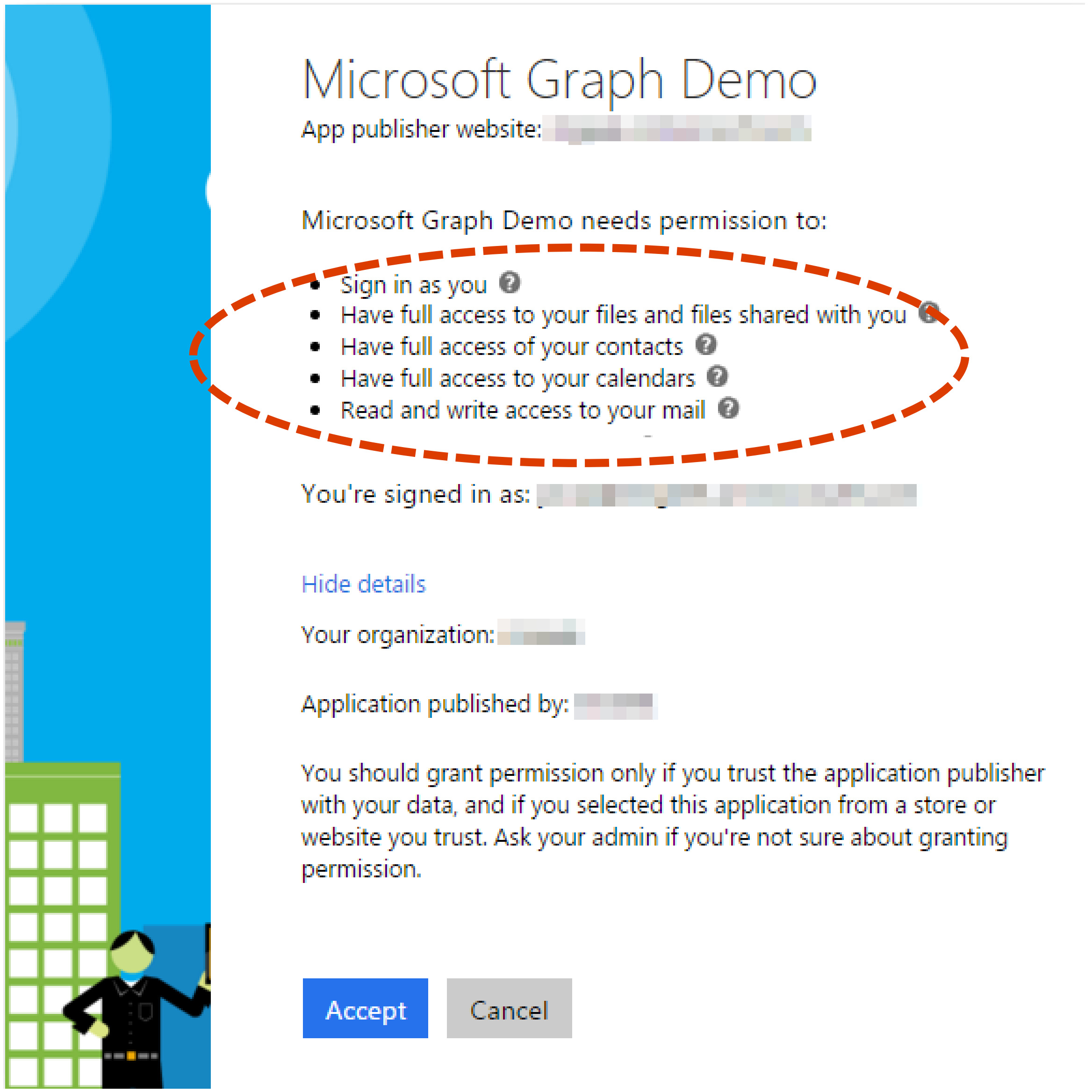
- Sign users in using OpenID Connect
  - Azure AD and Office 365 services
  - Supports MFA and federated user sign-in
- Device apps, web sites, SPAs, and service apps
- Pin apps to Office 365 app launcher from My apps



Office

Common consent

- Single auth flow for accessing all O365 services
- Admin and end-user consent
- Secure protocol
  - OpenID Connect and OAuth 2.0
  - No capturing user credentials
  - Fine-grained access scopes
  - Long-term access through refresh tokens



Microsoft Graph Demo

App publisher website: [redacted]

Microsoft Graph Demo needs permission to:

- Sign in as you
- Have full access to your files and files shared with you
- Have full access of your contacts
- Have full access to your calendars
- Read and write access to your mail

You're signed in as: [redacted]

Hide details

Your organization: [redacted]

Application published by: [redacted]

You should grant permission only if you trust the application publisher with your data, and if you selected this application from a store or website you trust. Ask your admin if you're not sure about granting permission.

Office

Authentication Options

Azure AD only

Separate auth flow supports Azure AD accounts only

Azure AD and Microsoft Accounts (Preview)

Converged auth flow supports Azure AD accounts and Microsoft accounts (LiveID - hotmail.com, etc.)

Office

Microsoft Account + Azure AD

Many apps want to sign users in from both Microsoft account and Azure AD

Now in preview:

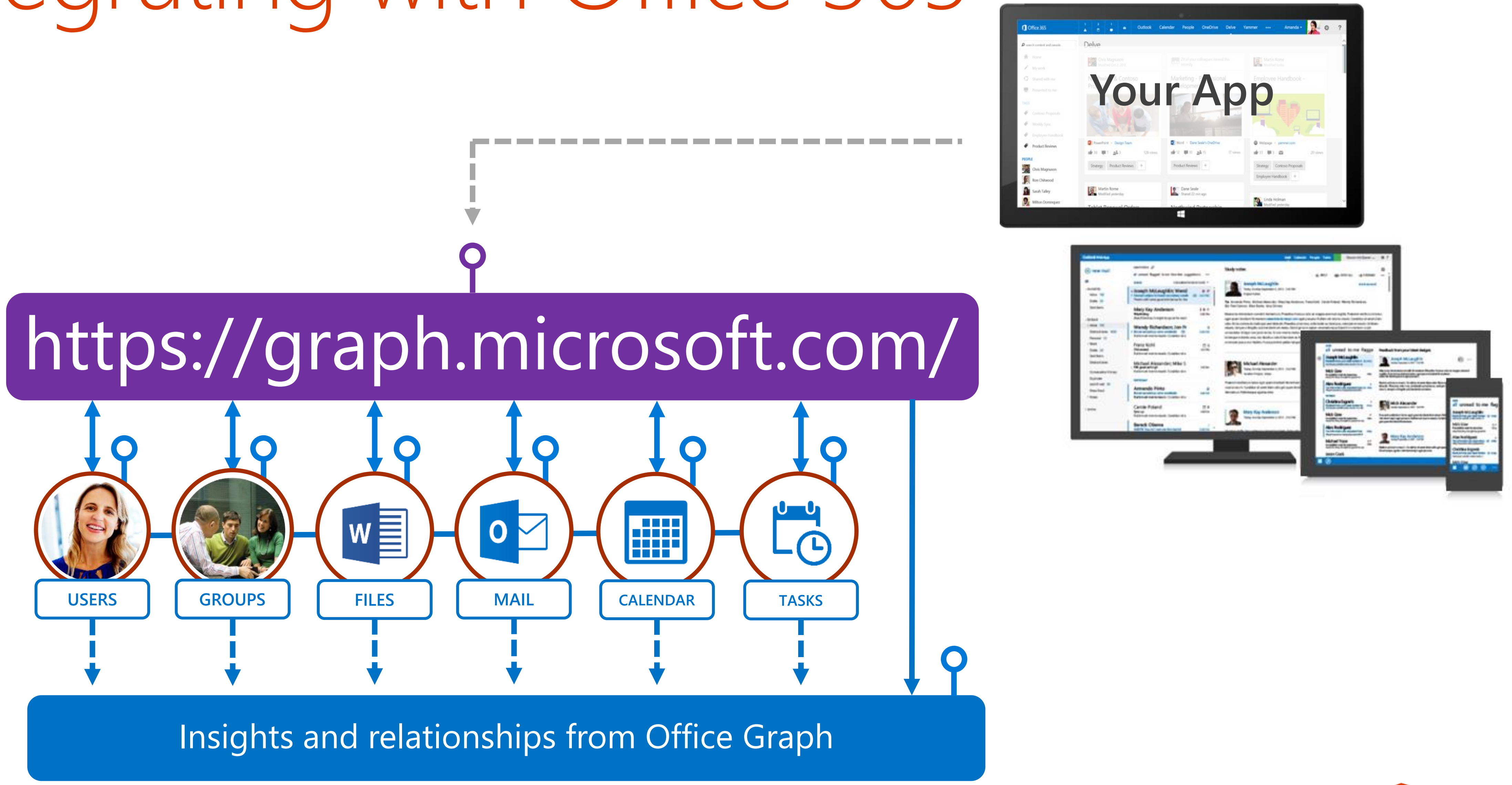
Single endpoint, OpenID Connect and OAuth 2.0
Single SDK
Single end user sign in experience
Single app registration experience

Works with Microsoft Graph

Single API endpoint, business and consumer data

Office

Integrating with Office 365



Your App
https://graph.microsoft.com/
USERS
GROUPS
FILES
MAIL
CALENDAR
TASKS
Insights and relationships from Office Graph

Office

Demo

Do.com
Store
Graph Visualizer
Graph Explorer

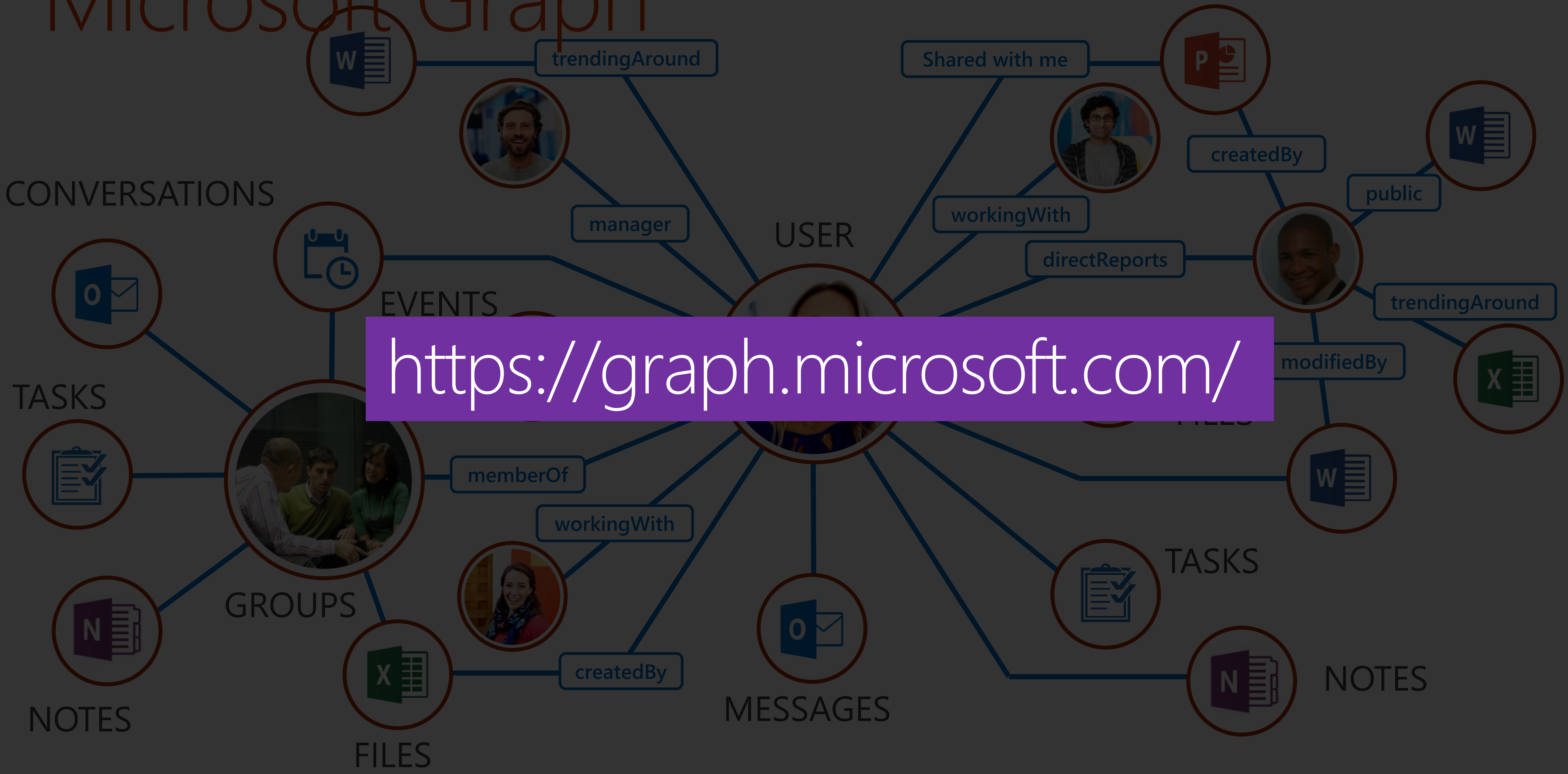
Office

2

Getting started

Office

Microsoft Graph



https://graph.microsoft.com/

Demo

Documentation
App registration

Office

3

SDKS and Code samples

Office 365

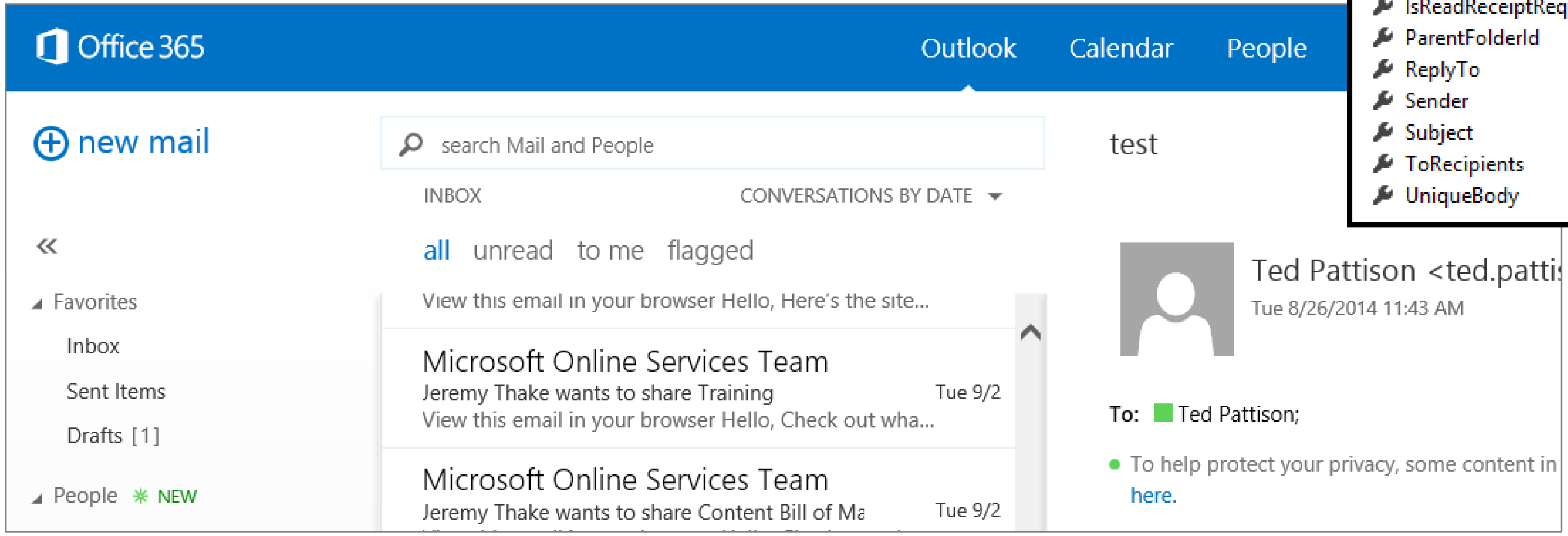
http://dev.office.com/

Office 365 APIs for Calendar, Mail and Contacts

- Office 365 APIs
  - Mail Message API
  - Calendar Events API
  - Contacts API
- Office 365 APIs accessible through REST
  - **https://outlook.office365.com/api/v1.0/me/messages**
  - **https://outlook.office365.com/api/v1.0/me/events**
  - **https://outlook.office365.com/api/v1.0/me/contacts**
- Office 365 APIs accessible through OutlookServicesClient
  - A library which abstracts away sending and receiving REST request

Mail Messages

- Common API operations
  - Reading messages
  - Deleting messages
  - Sending messages
  - Working with attachments



View this email in your browser Hello, Here's the site...

Microsoft Online Services Team
Jeremy Thake wants to share Training — Tue 9/2
View this email in your browser Hello, Check out wha...

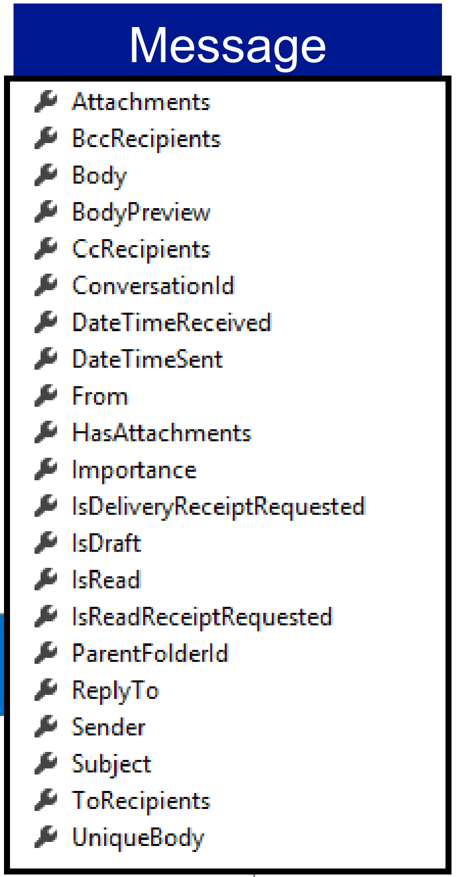
Microsoft Online Services Team
Jeremy Thake wants to share Content Bill of Ma — Tue 9/2

test

Ted Pattison <ted.patti
Tue 8/26/2014 11:43 AM

To: Ted Pattison;

To help protect your privacy, some content in
here.

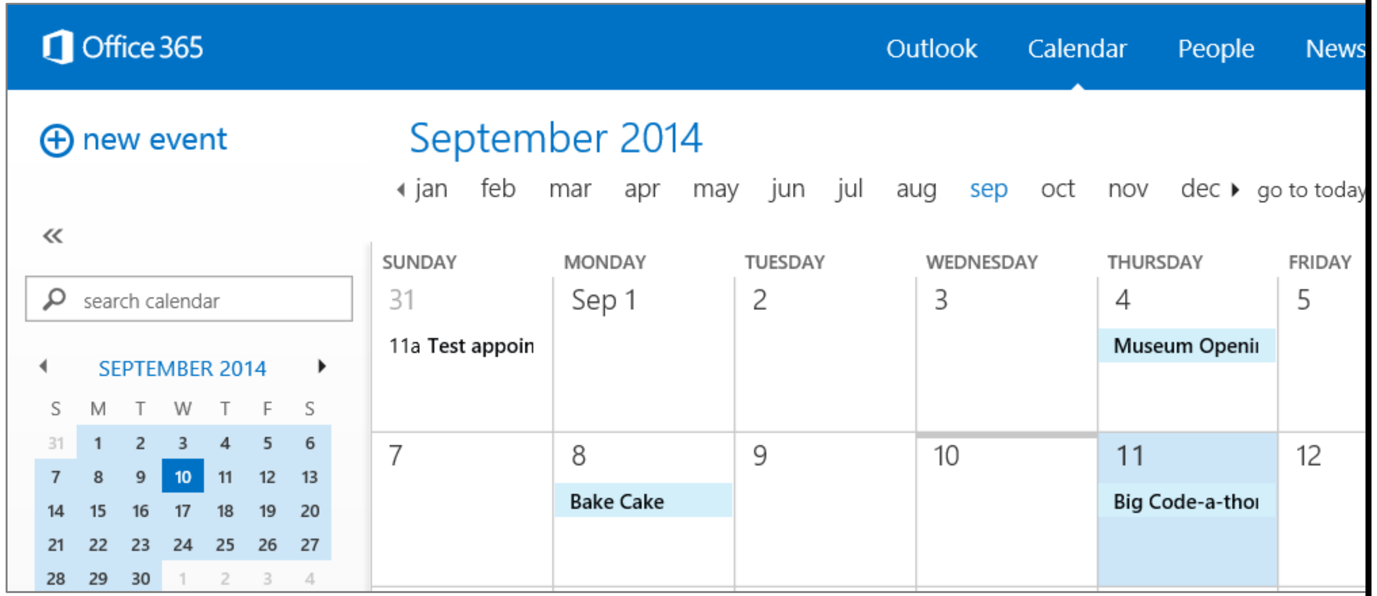


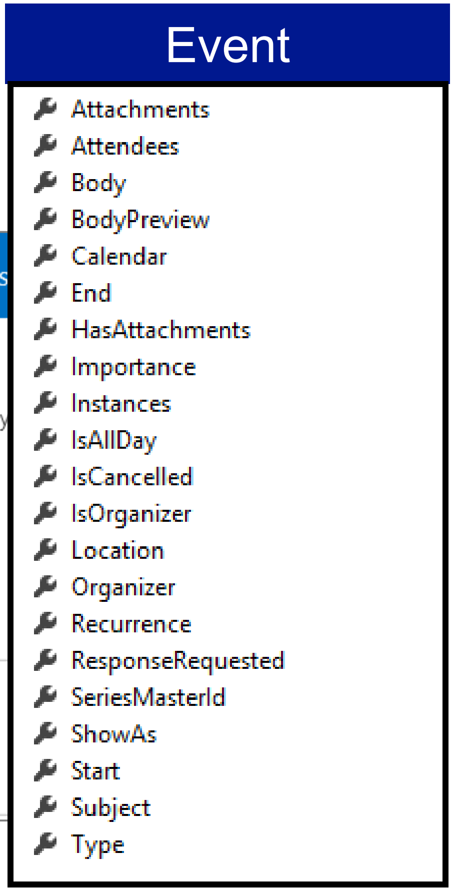
Message

- Attachments
- BccRecipients
- Body
- BodyPreview
- CcRecipients
- ConversationId
- DateTimeReceived
- DateTimeSent
- From
- HasAttachments
- Importance
- IsDeliveryReceiptRequested
- IsDraft
- IsRead
- IsReadReceiptRequested
- ParentFolderId
- ReplyTo
- Sender
- Subject
- ToRecipients
- UniqueBody

Calendar Events

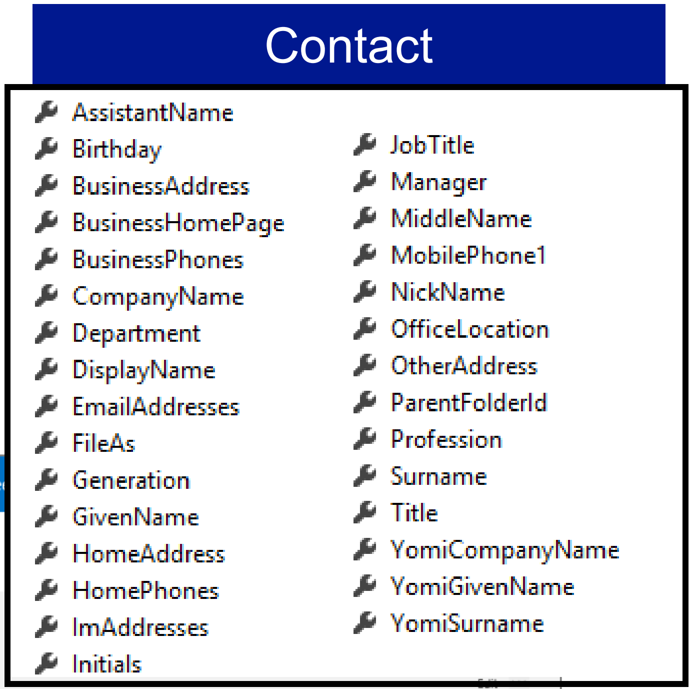
- Common API operations
  - Reading events for specific date range
  - Creating events
  - Deleting events
  - Editing events





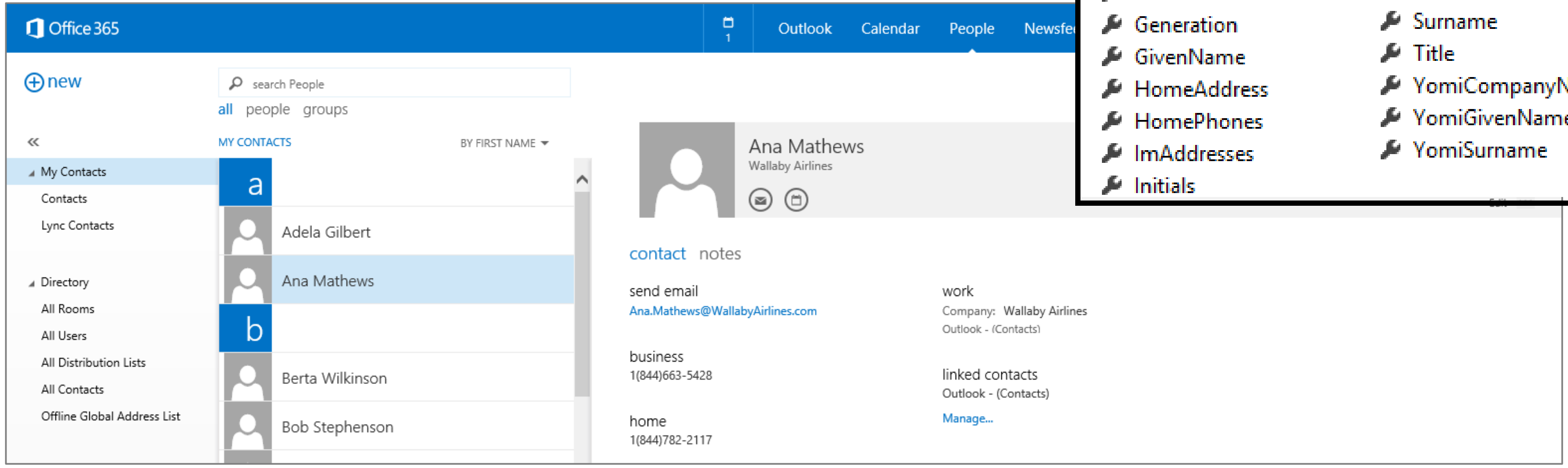
Contacts

- Common API operations
  - Reading contacts
  - Searching for contacts
  - Creating contacts
  - Deleting contacts
  - Editing events



Contact

- AssistantName
- Birthday
- BusinessAddress
- BusinessHomePage
- BusinessPhones
- CompanyName
- Department
- DisplayName
- EmailAddresses
- FileAs
- Generation
- GivenName
- HomeAddress
- HomePhones
- ImAddresses
- Initials
- JobTitle
- Manager
- MiddleName
- MobilePhone1
- NickName
- OfficeLocation
- OtherAddress
- ParentFolderId
- Profession
- Surname
- Title
- YomiCompanyName
- YomiGivenName
- YomiSurname



Adela Gilbert
Ana Mathews
Berta Wilkinson
Bob Stephenson

Ana Mathews
Wallaby Airlines

send email
Ana.Mathews@WallabyAirlines.com

business
1(844)663-5428

home
1(844)782-2117

work
Company: Wallaby Airlines
Outlook - (Contacts)

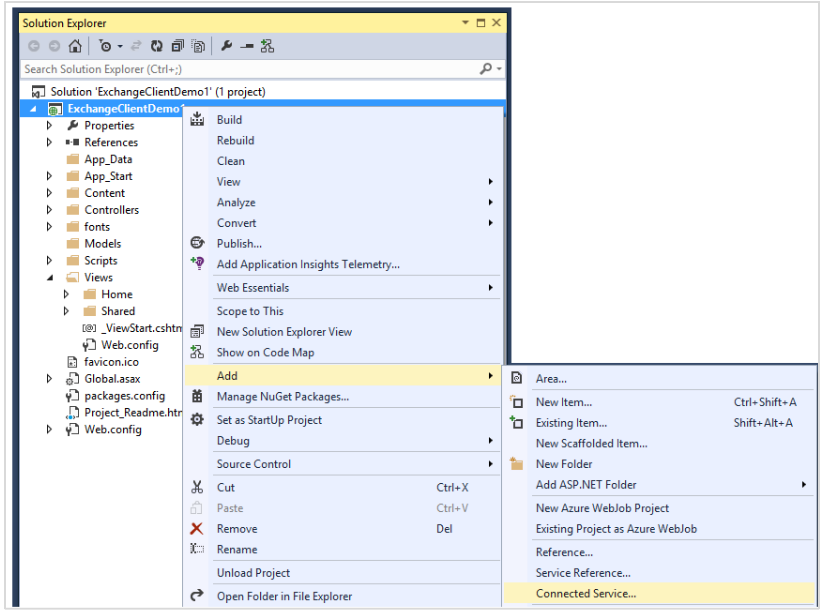
linked contacts
Outlook - (Contacts)

MVA Microsoft Virtual Academy

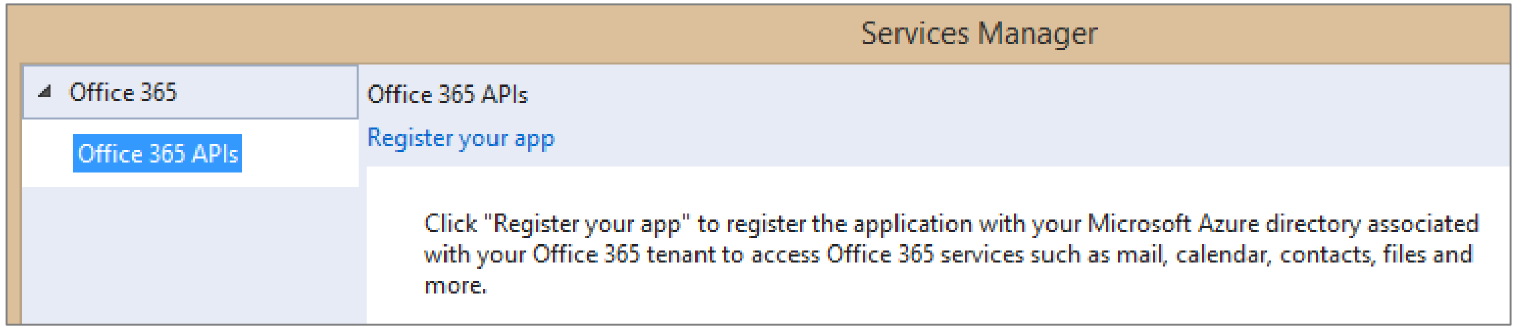
Connected Services

Adding Connected Services

1 Project > Add > Connected Service…



2 Register your app

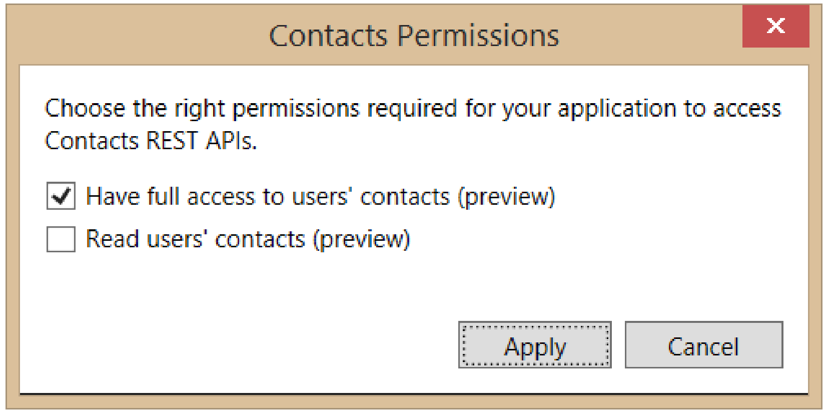


Click "Register your app" to register the application with your Microsoft Azure directory associated with your Office 365 tenant to access Office 365 services such as mail, calendar, contacts, files and more.

Office

Connected Services Permissions

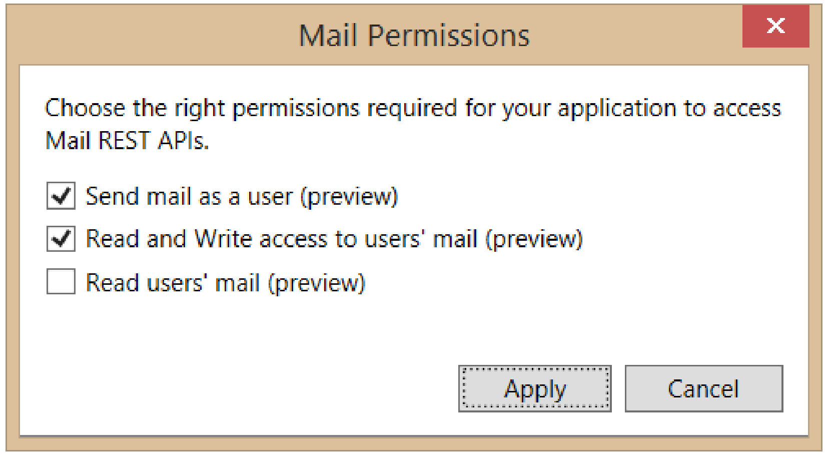
3 Select required app permissions



Contacts Permissions

Choose the right permissions required for your application to access Contacts REST APIs.

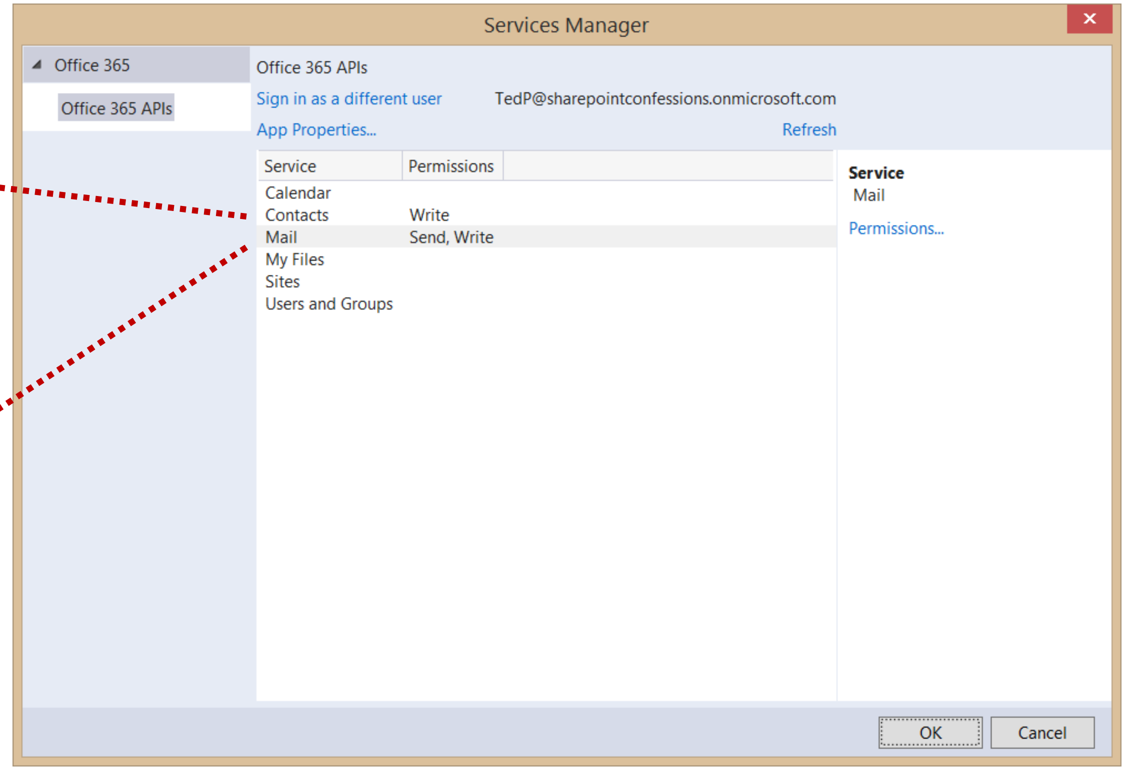
- [✓] Have full access to users' contacts (preview)
- [ ] Read users' contacts (preview)



Mail Permissions

Choose the right permissions required for your application to access Mail REST APIs.

- [✓] Send mail as a user (preview)
- [✓] Read and Write access to users' mail (preview)
- [ ] Read users' mail (preview)



Services Manager

Office 365 / Office 365 APIs

Office 365 APIs

Sign in as a different user TedP@sharepointconfessions.onmicrosoft.com

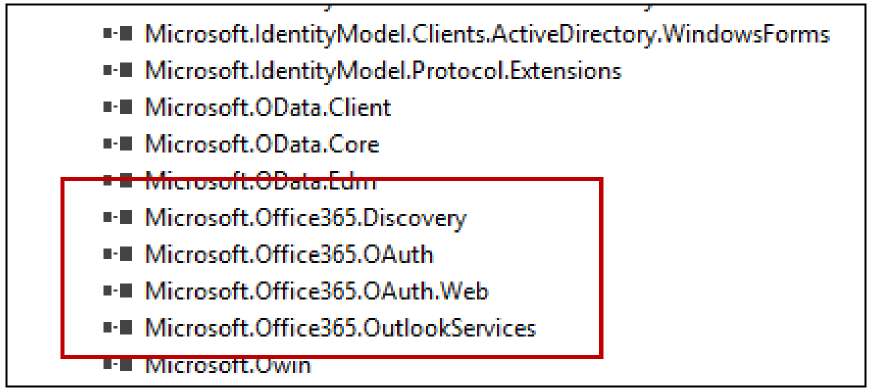
App Properties... Refresh

Service	Permissions
Calendar	
Contacts	Write
Mail	Send, Write
My Files	
Sites	
Users and Groups	

Service
Mail
Permissions...

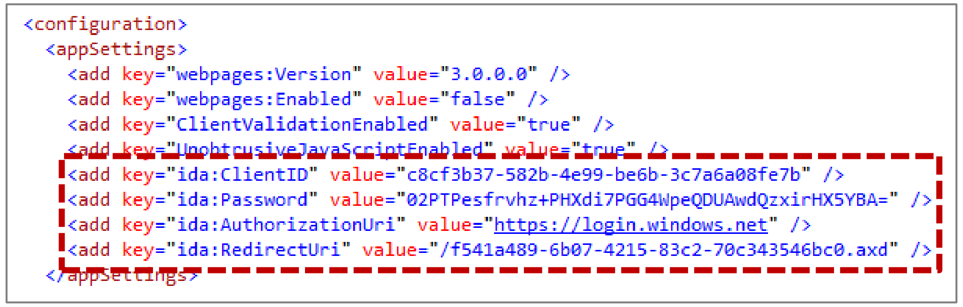
Projects with Connected Services

- Assemblies added with a Connected Service
  - Microsoft.Office365.OAuth
  - Microsoft.Office365.Oauth.Web
  - Microsoft.Office365.Exchange

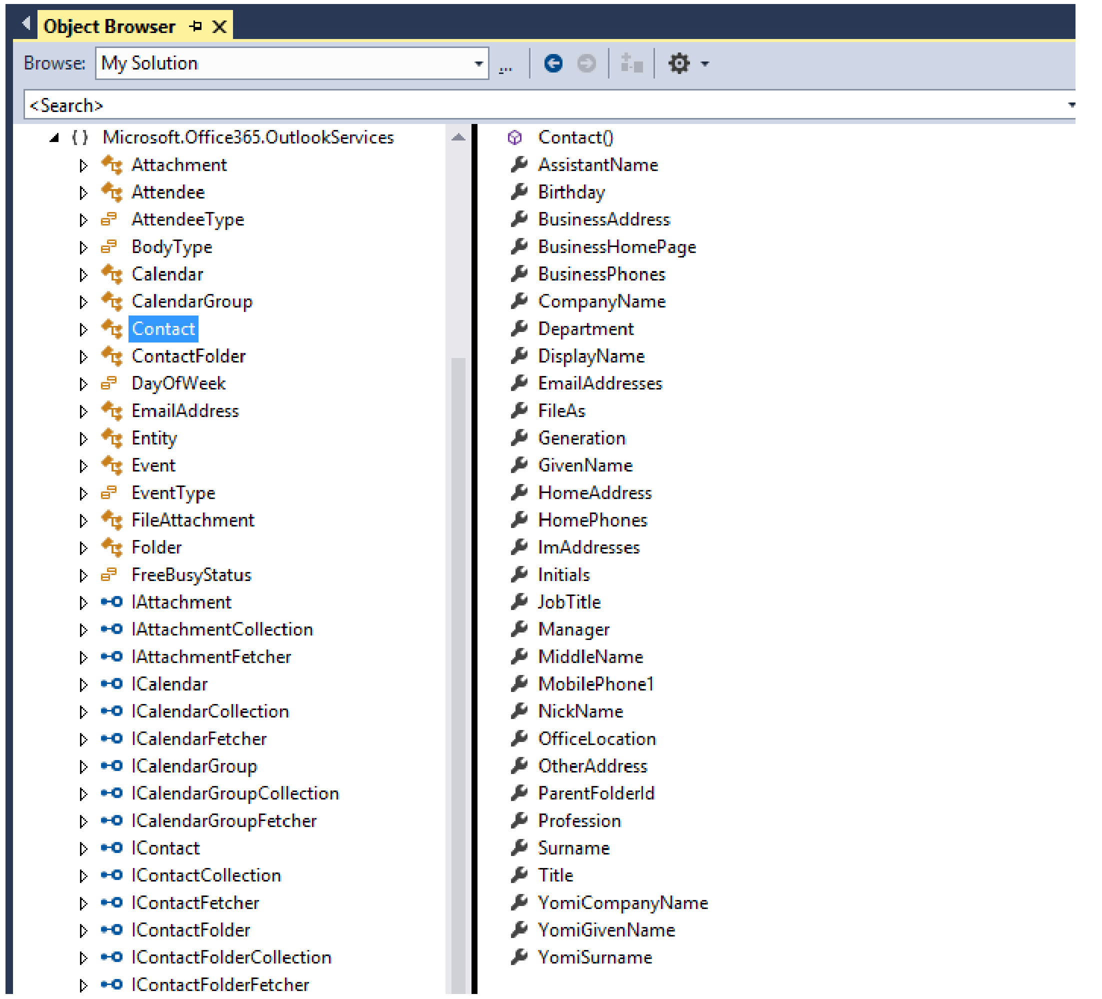


```
Microsoft.IdentityModel.Clients.ActiveDirectory.WindowsForms
Microsoft.IdentityModel.Protocol.Extensions
Microsoft.OData.Client
Microsoft.OData.Core
Microsoft.OData.Edm
Microsoft.Office365.Discovery
Microsoft.Office365.OAuth
Microsoft.Office365.OAuth.Web
Microsoft.Office365.OutlookServices
Microsoft.Owin
```

AppSettings added with a Connected Service



```
<configuration>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
    <add key="ida:ClientID" value="c8cf3b37-582b-4e99-be6b-3c7a6a08fe7b" />
    <add key="ida:Password" value="02PTPesfrvhz+PHXdi7PGG4WpeQDUAwdQzxirHX5YBA=" />
    <add key="ida:AuthorizationUri" value="https://login.windows.net" />
    <add key="ida:RedirectUri" value="/f541a489-6b07-4215-83c2-70c343546bc0.axd" />
  </appSettings>
```



Object Browser

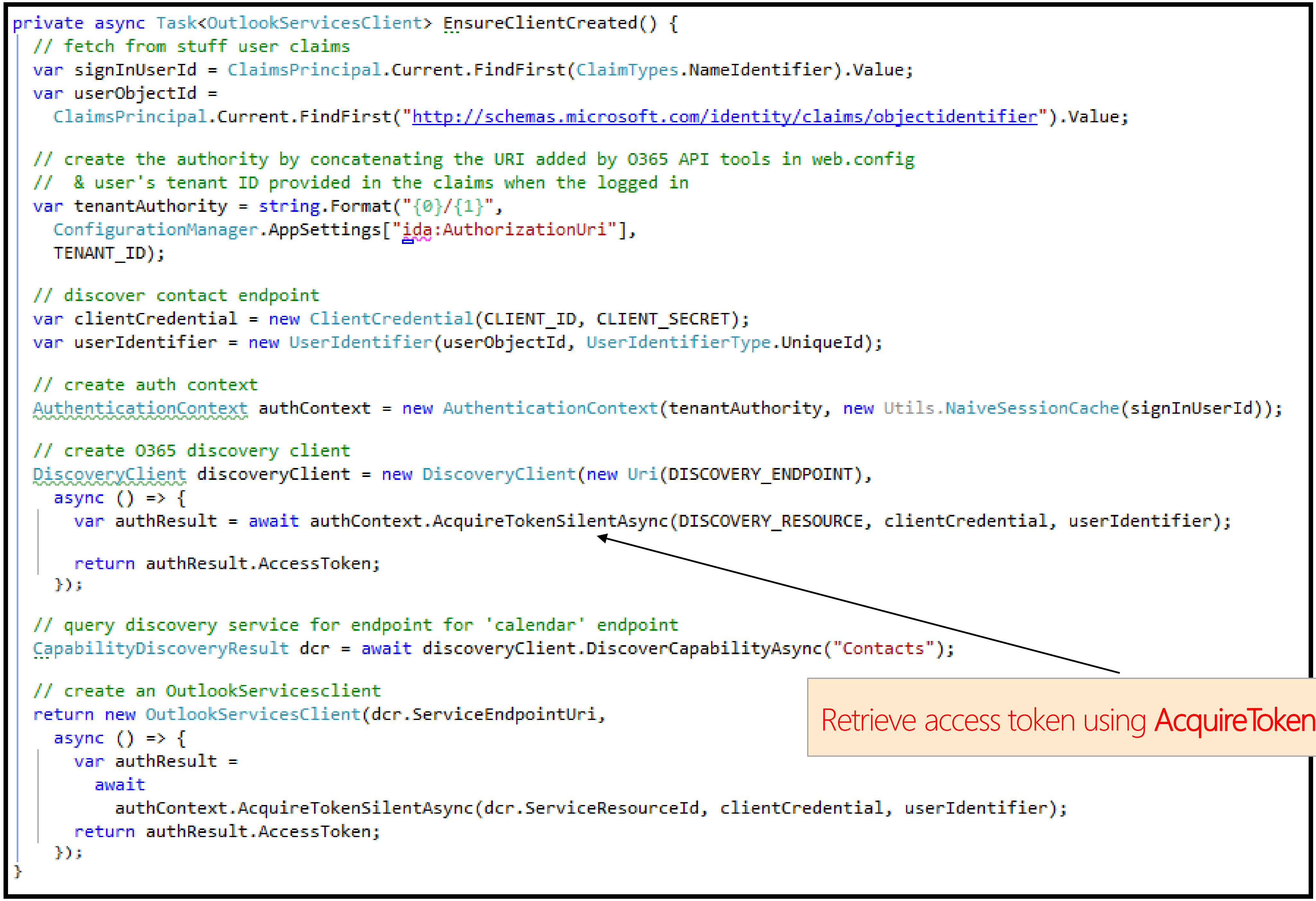
Browse: My Solution

<Search>

```
{ } Microsoft.Office365.OutlookServices
    Attachment
    Attendee
    AttendeeType
    BodyType
    Calendar
    CalendarGroup
    Contact
    ContactFolder
    DayOfWeek
    EmailAddress
    Entity
    Event
    EventType
    FileAttachment
    Folder
    FreeBusyStatus
    IAttachment
    IAttachmentCollection
    IAttachmentFetcher
    ICalendar
    ICalendarCollection
    ICalendarFetcher
    ICalendarGroup
    ICalendarGroupCollection
    ICalendarGroupFetcher
    IContact
    IContactCollection
    IContactFetcher
    IContactFolder
    IContactFolderCollection
    IContactFolderFetcher
```

```
Contact()
AssistantName
Birthday
BusinessAddress
BusinessHomePage
BusinessPhones
CompanyName
Department
DisplayName
EmailAddresses
FileAs
Generation
GivenName
HomeAddress
HomePhones
ImAddresses
Initials
JobTitle
Manager
MiddleName
MobilePhone1
NickName
OfficeLocation
OtherAddress
ParentFolderId
Profession
Surname
Title
YomiCompanyName
YomiGivenName
YomiSurname
```

Office

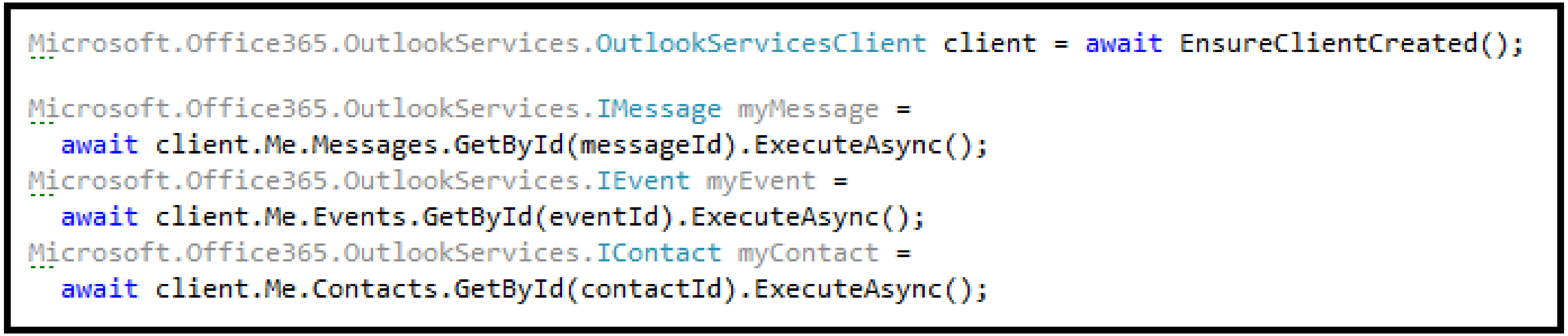


```
private async Task<OutlookServicesClient> EnsureClientCreated() {
  // fetch from stuff user claims
  var signInUserId = ClaimsPrincipal.Current.FindFirst(ClaimTypes.NameIdentifier).Value;
  var userObjectId =
    ClaimsPrincipal.Current.FindFirst("http://schemas.microsoft.com/identity/claims/objectidentifier").Value;

  // create the authority by concatenating the URI added by O365 API tools in web.config
  //  & user's tenant ID provided in the claims when the logged in
  var tenantAuthority = string.Format("{0}/{1}",
    ConfigurationManager.AppSettings["ida:AuthorizationUri"],
    TENANT_ID);

  // discover contact endpoint
  var clientCredential = new ClientCredential(CLIENT_ID, CLIENT_SECRET);
  var userIdentifier = new UserIdentifier(userObjectId, UserIdentifierType.UniqueId);

  // create auth context
  AuthenticationContext authContext = new AuthenticationContext(tenantAuthority, new Utils.NaiveSessionCache(signInUserId));

  // create O365 discovery client
  DiscoveryClient discoveryClient = new DiscoveryClient(new Uri(DISCOVERY_ENDPOINT),
    async () => {
      var authResult = await authContext.AcquireTokenSilentAsync(DISCOVERY_RESOURCE, clientCredential, userIdentifier);

      return authResult.AccessToken;
    });

  // query discovery service for endpoint for 'calendar' endpoint
  CapabilityDiscoveryResult dcr = await discoveryClient.DiscoverCapabilityAsync("Contacts");

  // create an OutlookServicesclient
  return new OutlookServicesClient(dcr.ServiceEndpointUri,
    async () => {
      var authResult =
        await
          authContext.AcquireTokenSilentAsync(dcr.ServiceResourceId, clientCredential, userIdentifier);
      return authResult.AccessToken;
    });
}
```

Retrieve access token using **AcquireTokenSlientAsync**

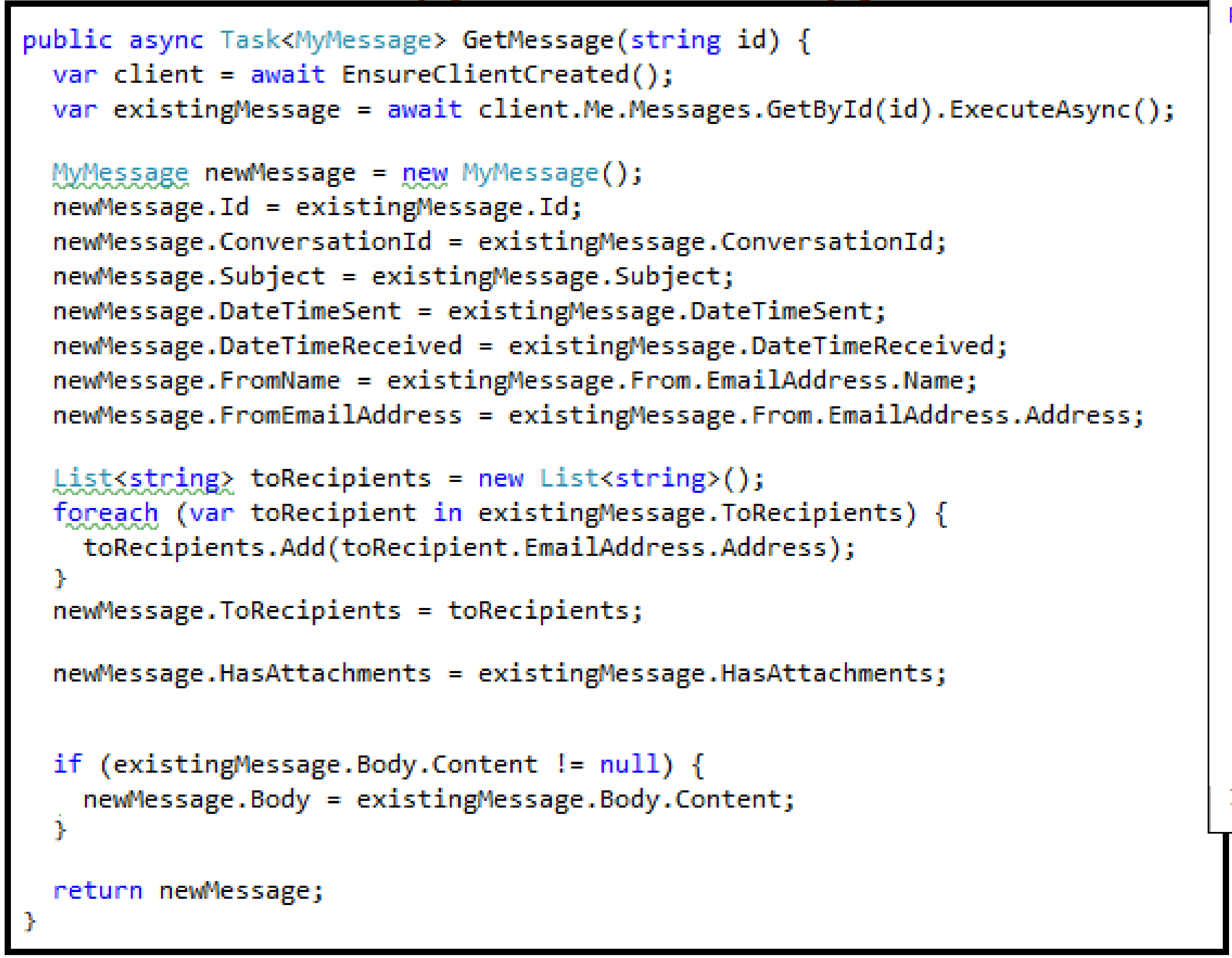
Programming with OutlookServiceClient

- OutlookServiceClient provide **Me** property
  - Provides access to mail, events and contacts of currently logged in user
  - Explicit calls used (e.g. **ExecuteAsync**) to call across network to Office 365 service

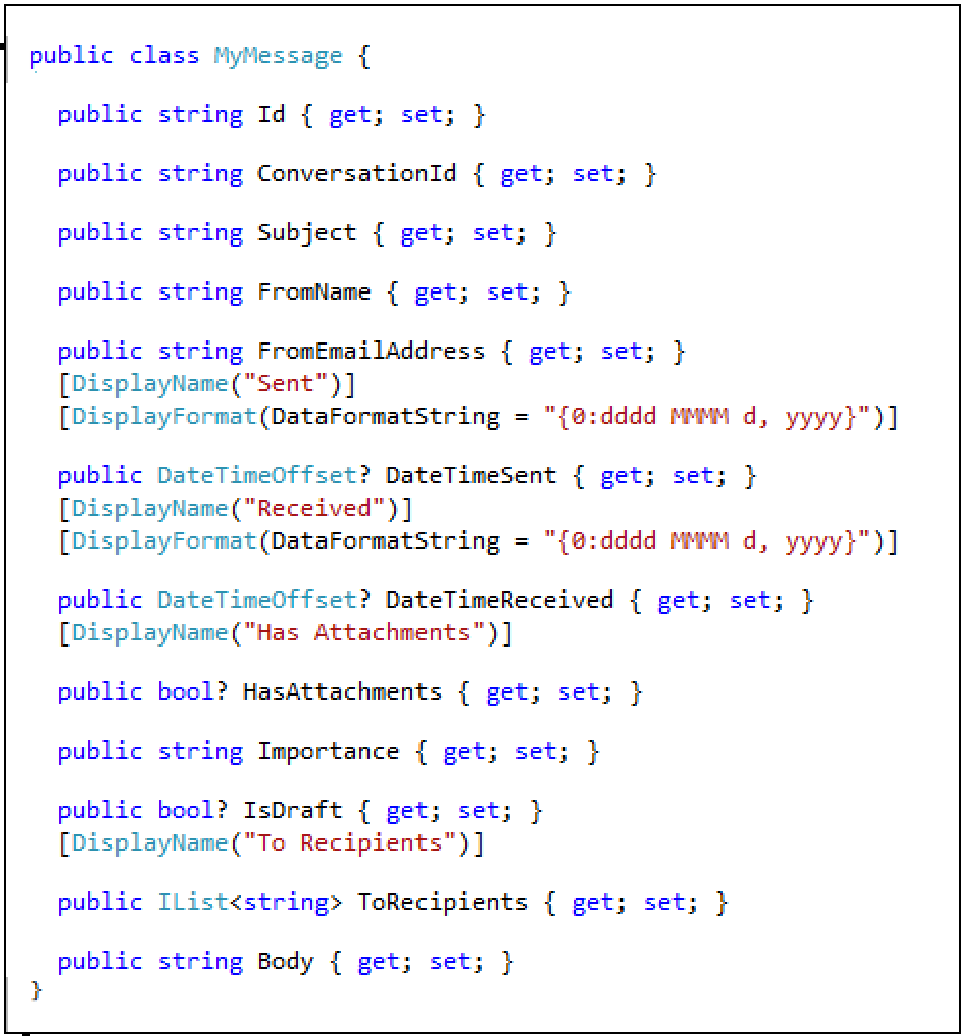


```
Microsoft.Office365.OutlookServices.OutlookServicesClient client = await EnsureClientCreated();

Microsoft.Office365.OutlookServices.IMessage myMessage =
  await client.Me.Messages.GetById(messageId).ExecuteAsync();
Microsoft.Office365.OutlookServices.IEvent myEvent =
  await client.Me.Events.GetById(eventId).ExecuteAsync();
Microsoft.Office365.OutlookServices.IContact myContact =
  await client.Me.Contacts.GetById(contactId).ExecuteAsync();
```

Retrieving Messages



```
public async Task<MyMessage> GetMessage(string id) {
  var client = await EnsureClientCreated();
  var existingMessage = await client.Me.Messages.GetById(id).ExecuteAsync();

  MyMessage newMessage = new MyMessage();
  newMessage.Id = existingMessage.Id;
  newMessage.ConversationId = existingMessage.ConversationId;
  newMessage.Subject = existingMessage.Subject;
  newMessage.DateTimeSent = existingMessage.DateTimeSent;
  newMessage.DateTimeReceived = existingMessage.DateTimeReceived;
  newMessage.FromName = existingMessage.From.EmailAddress.Name;
  newMessage.FromEmailAddress = existingMessage.From.EmailAddress.Address;

  List<string> toRecipients = new List<string>();
  foreach (var toRecipient in existingMessage.ToRecipients) {
    toRecipients.Add(toRecipient.EmailAddress.Address);
  }
  newMessage.ToRecipients = toRecipients;

  newMessage.HasAttachments = existingMessage.HasAttachments;


  if (existingMessage.Body.Content != null) {
    newMessage.Body = existingMessage.Body.Content;
  }

  return newMessage;
}
```

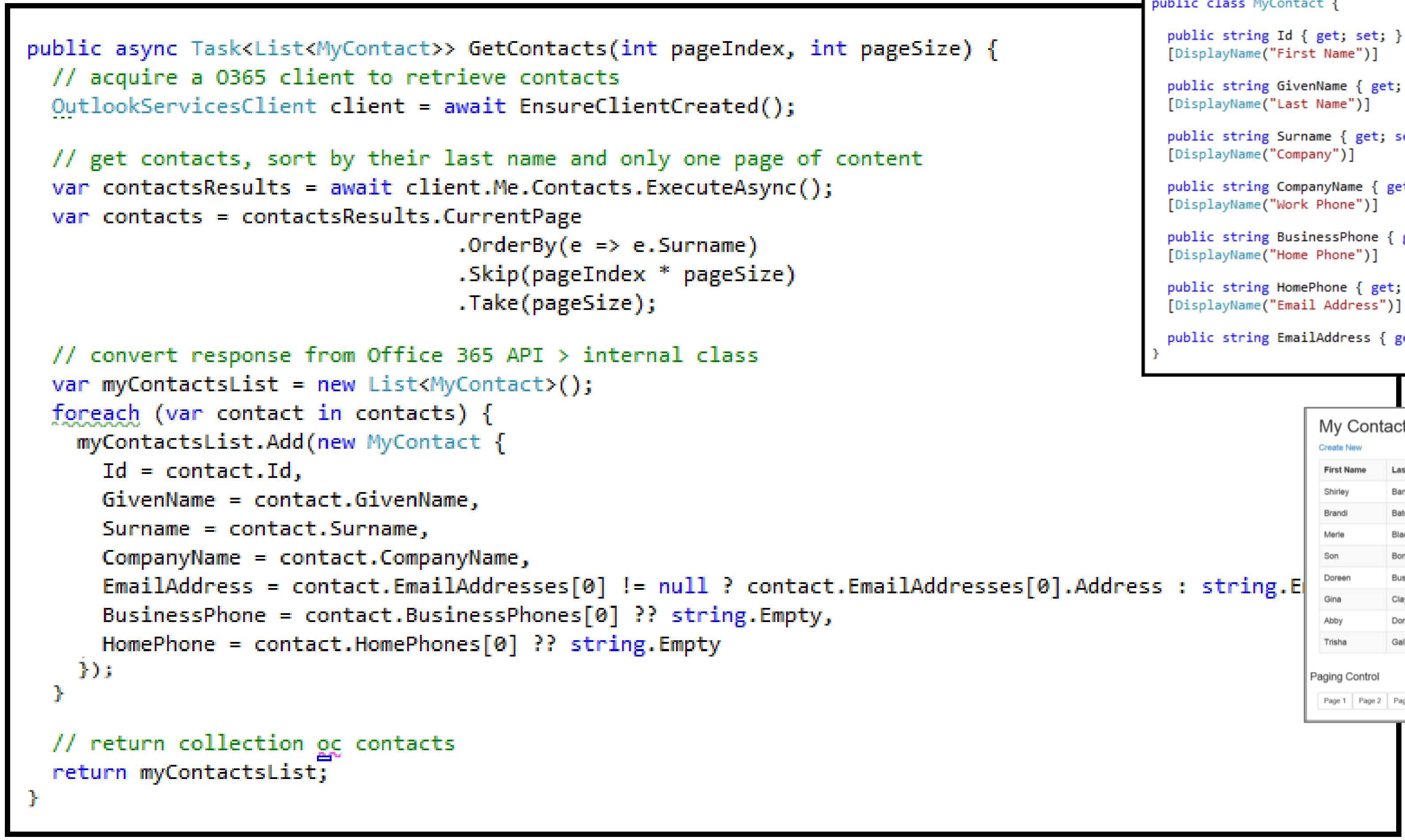


```
public class MyMessage {

  public string Id { get; set; }

  public string ConversationId { get; set; }

  public string Subject { get; set; }

  public string FromName { get; set; }

  public string FromEmailAddress { get; set; }
  [DisplayName("Sent")]
  [DisplayFormat(DataFormatString = "{0:dddd MMMM d, yyyy}")]

  public DateTimeOffset? DateTimeSent { get; set; }
  [DisplayName("Received")]
  [DisplayFormat(DataFormatString = "{0:dddd MMMM d, yyyy}")]

  public DateTimeOffset? DateTimeReceived { get; set; }
  [DisplayName("Has Attachments")]

  public bool? HasAttachments { get; set; }

  public string Importance { get; set; }

  public bool? IsDraft { get; set; }
  [DisplayName("To Recipients")]

  public IList<string> ToRecipients { get; set; }

  public string Body { get; set; }
}
```

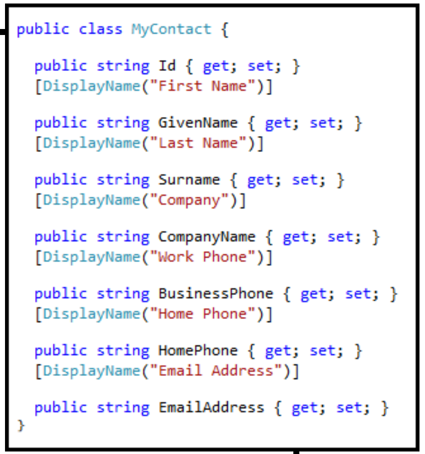
Office

Paging with the Office 365 APIs

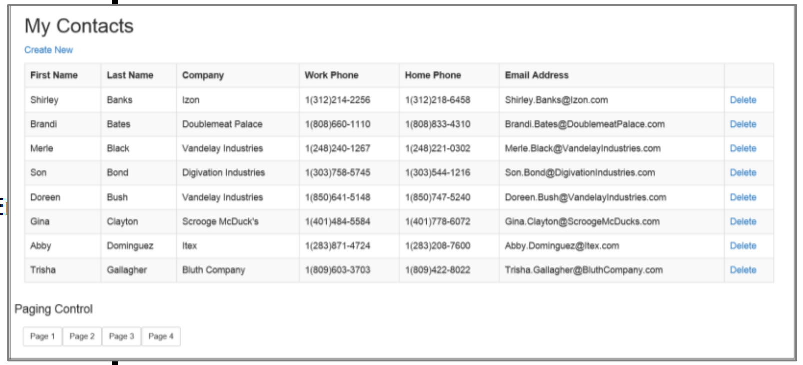
- Limit the number of items returned by using the Take() and Skip() prior to ExecuteAsync()
- Paging buttons call GetContacts which skips over the specified number of contacts to present the next page



```
public async Task<List<MyContact>> GetContacts(int pageIndex, int pageSize) {
  // acquire a O365 client to retrieve contacts
  OutlookServicesClient client = await EnsureClientCreated();

  // get contacts, sort by their last name and only one page of content
  var contactsResults = await client.Me.Contacts.ExecuteAsync();
  var contacts = contactsResults.CurrentPage
                                .OrderBy(e => e.Surname)
                                .Skip(pageIndex * pageSize)
                                .Take(pageSize);

  // convert response from Office 365 API > internal class
  var myContactsList = new List<MyContact>();
  foreach (var contact in contacts) {
    myContactsList.Add(new MyContact {
      Id = contact.Id,
      GivenName = contact.GivenName,
      Surname = contact.Surname,
      CompanyName = contact.CompanyName,
      EmailAddress = contact.EmailAddresses[0] != null ? contact.EmailAddresses[0].Address : string.E
      BusinessPhone = contact.BusinessPhones[0] ?? string.Empty,
      HomePhone = contact.HomePhones[0] ?? string.Empty
    });
  }

  // return collection oc contacts
  return myContactsList;
}
```



```
public class MyContact {

  public string Id { get; set; }
  [DisplayName("First Name")]

  public string GivenName { get; set; }
  [DisplayName("Last Name")]

  public string Surname { get; set; }
  [DisplayName("Company")]

  public string CompanyName { get; set; }
  [DisplayName("Work Phone")]

  public string BusinessPhone { get; set; }
  [DisplayName("Home Phone")]

  public string HomePhone { get; set; }
  [DisplayName("Email Address")]

  public string EmailAddress { get; set; }
}
```



My Contacts

Create New

First Name	Last Name	Company	Work Phone	Home Phone	Email Address	
Shirley	Banks	Izon	1(312)214-2256	1(312)218-6458	Shirley.Banks@Izon.com	Delete
Brandi	Bates	Doublemeat Palace	1(808)660-1110	1(808)833-4310	Brandi.Bates@DoublemeatPalace.com	Delete
Merle	Black	Vandelay Industries	1(248)240-1267	1(248)221-0302	Merle.Black@VandelayIndustries.com	Delete
Son	Bond	Digivation Industries	1(303)758-5745	1(303)544-1216	Son.Bond@DigivationIndustries.com	Delete
Doreen	Bush	Vandelay Industries	1(850)641-5148	1(850)747-5240	Doreen.Bush@VandelayIndustries.com	Delete
Gina	Clayton	Scrooge McDuck's	1(401)484-5584	1(401)778-6072	Gina.Clayton@ScroogeMcDucks.com	Delete
Abby	Dominguez	Itex	1(283)871-4724	1(283)208-7600	Abby.Dominguez@Itex.com	Delete
Trisha	Gallagher	Bluth Company	1(809)603-3703	1(809)422-8022	Trisha.Gallagher@BluthCompany.com	Delete

Paging Control

Page 1 Page 2 Page 3 Page 4

Demo

SDKs

Code samples

Office

4

Resources

Office

http://dev.office.com/

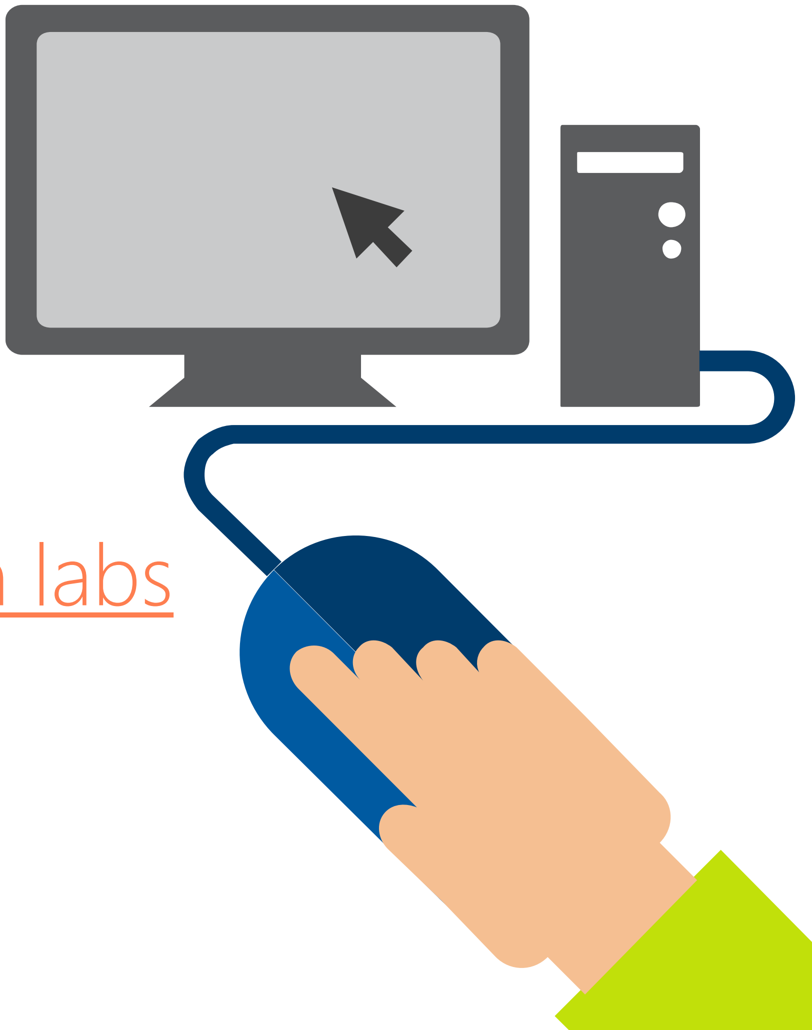
Resources

- Documentation, samples and more:
  - https://graph.microsoft.com/
- Stack overflow:

  [MicrosoftGraph] and [Office365]
- Twitter:

  #MicrosoftGraph and #Office365dev

Further reading…



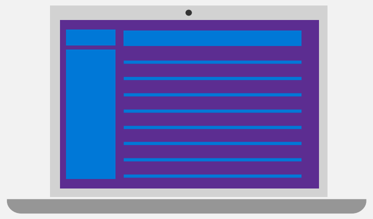
[Getting Started with APIs]

[Microsoft Graph API Code Samples]

[Microsoft Graph API Training videos & hands on labs]

[Microsoft Graph API documentation]

Developer Program

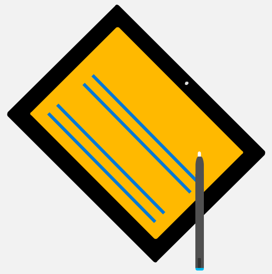


E-mail
Newsletters

Free
Developer
Subscription

Free
Training

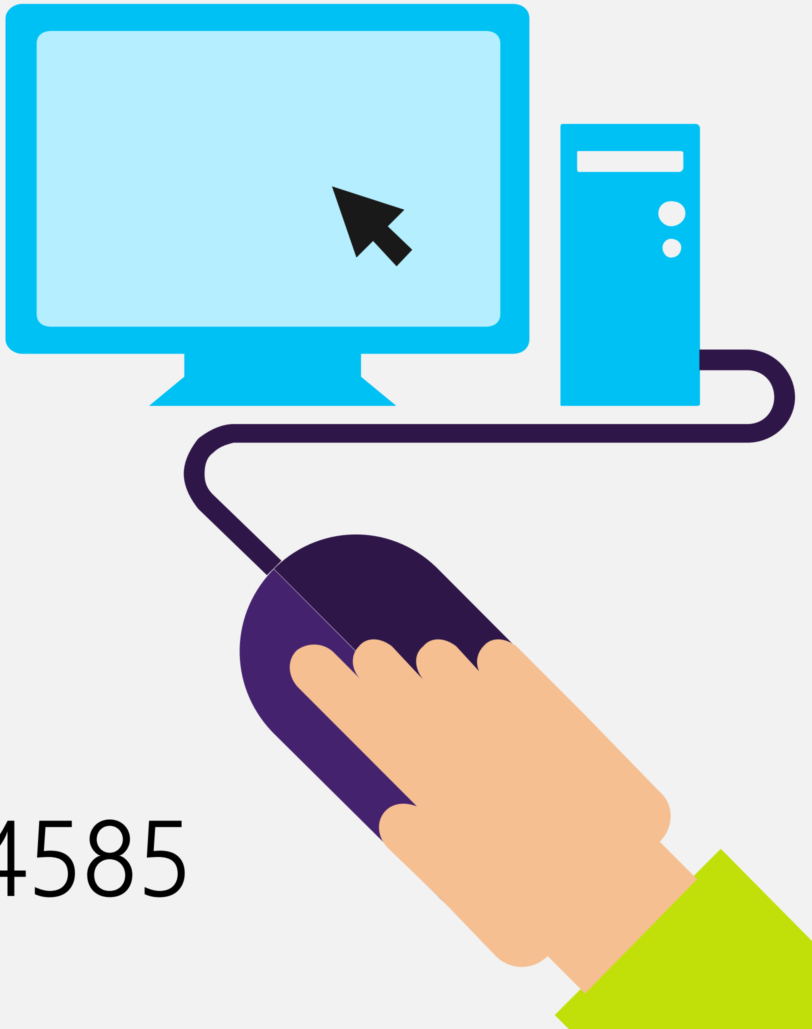
Free
Tools



Webinars

http://dev.[illegible]program

HOL

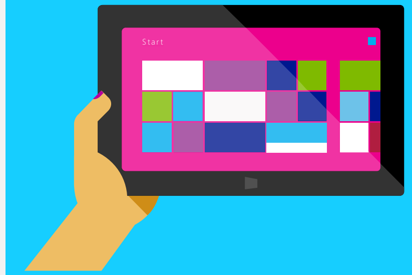


https://dev.office.com/hands-on-labs/4585

Engage

Office 365 Network

https://www.yammer.com/itpronetwork



Twitter

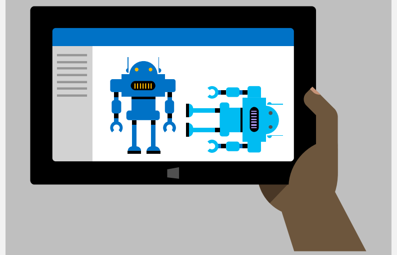
@OfficeDev

Stack overflow

[ms-office]

Channel 9
Dev Show

http://aka.ms/O365DevShow



Podcasts

http://dev.office.com/podcasts

Snack Demos

http://aka.ms/o365DevSnackDemos

UserVoice

http://officespdev.uservoice.com/

Microsoft