Consuming
Office 365 REST API

Paolo Pialorsi – paolo@pialorsi.com – PiaSys.com
About me

• Project Manager, Consultant, Trainer
• About 50 Microsoft certification exams passed, including MC(S)M
• MVP Office 365
• Focused on SharePoint since 2002
• Author of 10 books about XML, SOAP, .NET, LINQ, and SharePoint
• Speaker at main IT conferences

@PaoloPia - paolo@pialorsi.com - http://www.pialorsi.com/blog.aspx
Introducing Office 365 REST API
What are the Office 365 REST API?

- Set of services with REST (REpresentational State Transfer) endpoints
- Available services
  - Microsoft Exchange Online
    - Mail, Contacts, Calendars
  - Microsoft OneDrive for Business
    - My Files
  - Microsoft SharePoint Online
    - Sites
  - Microsoft Azure Active Directory
    - Authentication, Directory Graph
The SharePoint client APIs

Sites, Lists and Libs | Workflow | Search | Taxonomy | BCS | ...

_api

Execute query

JavaScript library
Silverlight library
.Net CLR library

Custom client code

REST/OData
Office 365 REST APIs

**Active Directory**
- Users
- Groups

**Exchange & Outlook.com**
- Mail
- Calendar
- Contacts

**SharePoint**
- Sites
- Client API
  - Sites, Lists and Libs
  - Workflow
  - Search
  - Taxonomy
  - BCS
  - ...

**OneDrive**
- OneDrive
- OneDrive for Business
How to consume the APIs?

• Directly via REST endpoints
• Indirectly via high-level client libraries
  • .NET client libraries
  • JavaScript client libraries
  • Open Source SDKs for iOS and Android
• Supported platforms for .NET client libraries
  • .NET Windows Store Apps
  • Windows Forms Application
  • WPF Application
  • ASP.NET Web Forms/MVC
  • Xamarin Android and iOS Applications
  • Multi-device hybrid apps
DEMO

Playing with the APIs using Fiddler
.NET Environment Configuration

- Microsoft Visual Studio 2013
- Microsoft Office Developer Tools for Visual Studio 2013
- A Microsoft Office 365 tenant (can be a developer tenant)
- Some NuGet packages

- OWIN OpenId Connect
  - For ASP.NET only
DEMO

Preparing a development environment
Understanding Azure Active Directory
At the basis of everything

- There is Microsoft Azure Active Directory
  - Fundamental for the security architecture of Office 365
  - Useful by itself, even without Office 365
- To manage user’s identities
- To manage applications, their permissions, and their assignments
- To make the AD graph available
Microsoft Azure Active Directory

- On-Premises Directory
- DirSync
- OAuth 2.0
- SAML-P
- WS-Federation
- Federation Metadata
- Graph API
Application registration in Azure AD

• Before consuming Office 365 API you need to register and authorize applications
  • Can be done by Azure AD Admin UI
  • Can be done via REST API, as well (nice! 😊)

• Kind of applications
  • Native application
  • Web/REST API application

• Authorization protocol: OAuth 2.0

• Can be done automatically through Visual Studio 2013 and Office Developer Tools for Visual Studio 2013
  • Add -> “Connected Services”
Applications Permissions

• Native Applications
  • Desktop/mobile applications
  • Based on application’s ClientID and user’s credentials
  • End users will grant permissions to the application to act on her/his own behalf (Delegated Permissions)

• Web Applications
  • Web or REST API applications
  • Based on application’s ClientID and SharedSecret (Application Permissions)
  • Or based also on current user’s identity, as well (Delegated Permissions)
Multi-tenancy

• You can define an application to support multi-tenancy
  • It will be available in multiple Azure AD (Office 365) tenants
• You need to provide a sign-up process
• The client libraries support multi-tenant scenarios
  • You will see shortly ...
DEMO

Azure AD and how to register an application in Azure AD manually or via VS2013
Consuming Services
Steps to consume a service

• Authenticate against Azure AD
• Discover the service endpoint
• Get (or refresh) an OAuth Access Token
• Contact the endpoint
  • Providing the Access Token
Azure AD Authentication (Library)

• You can use Azure Active Directory Authentication Library (ADAL)
  • Available via NuGet
  • Provides AuthenticationContext and AuthenticationResult types, and some others ...
  • Useful to authenticate against Azure AD or local AD (ADFS 3.0)

• Supported Scenarios
  • Authenticating Users of a Client Application to a Remote Resource
  • Authenticating a Server Application to a Remote Resource
  • Authenticating a Server Application on Behalf of a User to Access a Remote Resource

• Leverages a TokenCache object
  • By default stores issued tokens within a native or a custom cache

• Provides automatic token refresh capabilities
Service Discovery

• Leverages a specific discovery REST service
  • https://api.office.com/discovery/v1.0/me
  • https://api.office.com/discovery/v1.0/me/AllServices

• There is a .NET client library
  • Available as a NuGet package
  • Includes *DiscoveryClient* type, and some others ...
  • Returns *ServiceResourceId* and *ServiceEnpointUri*
    • Based on the capability name
Invoking the Service

• The access token is acquired using the ADAL AuthenticationContext

• Create the client object based on the service URI and the access token

• Available client types
  • SharePointClient: SharePoint and OneDrive for Business
  • OutlookServicesClient: Mail, Calendar, Contacts
  • ActiveDirectoryClient: Azure AD Graph API
DEMO

Using the Office 365 REST API from a Windows desktop WPF application
DEMO

Using the Office 365 REST API from an ASP.NET MVC web application
Thanks!

Feedbacks are welcome: paolo@pialorsi.com

Code Samples:  
https://github.com/OfficeDev/PnP/
https://github.com/OfficeDev/PnP/tree/master/Samples/Office365Api.Overview