Office 365:

The best recipes for developers

Gustavo Velez

BOOK ONLY
(no Subscription)

güitaca publishers

First Edition

Office 365 - The best recipes for developers

First published: April 2020

ISBN 978-958-48-8932-4

Amsterdam - Bogotá - Bilbao

About Güitaca Publishers

Güitaca was an extremely beautiful goddess who celebrated a life full of joy, games, and pleasure. As a result of protesting against Bochica, the heavens CEO, she was turned into a white owl. Güitaca, coming from the Muisca mythology, is our symbol and guardian. The Muiscas were the pivot civilization of the Americas, between the Incas in the south, and the Aztecs in the north, inhabiting the Altiplano Cundiboyacense, a high plateau in the Colombian Andes.

Güitaca Publishers is an independent company aimed at bringing out technical books in electronic format, especially about Information Technology, computer developing, and systems engineering. We aim to present the most recent publications to our readers using novel processes as, for example, our "Books by subscription" concept.

Our team is internationally located and makes use of the most modern computer technology to communicate and produce our books. The Güitaca Publishers team is composed of:

Vicky Santana - Editor in Chief

Our CEO, with an ample experience managing international magazines and publications, Vicky inspires, administers, and coordinates the Güitaca enterprise. A journalist by profession and photograph by spirit, she takes care of the complete publishing process.

Alcira Blanco - Language Department

Alcira oversees our translation and editing department. She ensures that you can understand what our authors try to tell you. From her background in biology stems her affinity for environmental themes, and she is a translator and proofreader for the love of art and languages.

Martha Sarmiento - Graphic Design

You can read our books thanks to the designs made by Martha. Professional in graphic design, specialized in editorial processes, design and layout of print media, Martha has quite a long experience in the field.

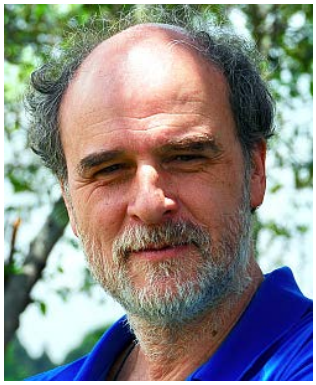
Alejandro Pérez & Jordi Niubó - Artwork Design

Alejandro and Jordi are the creators of our covers, logos and all the artwork of Güitaca. Alejandro has developed his career as copywriter, account executive and project coordinator in his bilbaosolutions.com own company. Jordi is trained in artistic disciplines and computer technologies (jordiniubo.com), with mentions of his work by the EIDE.

If you wish to make contact with the Güitaca Publishers team, please send us an email to info@guitaca.com. We are interested as well if you want to become one of our authors.

Amsterdam - Bogotá - Bilbao

About the author



Gustavo Velez is a mechanical and electronics engineer working as a software developer and senior solutions architect for more than thirty years. Specialized in integration of Microsoft software, he started working with SharePoint before the server got its current name: in 1998, Gustavo finalized his first enterprise collaboration project using Site Server, the precursor of SharePoint, and Microsoft's first effort at providing a solution to the growing business of Internet-based Document Management, Content Management, and Site Personalization. Gustavo is Microsoft's Most Valuable Professional (MVP) since 2008. In his many years of experience developing and working with Windows and Office applications, Gustavo has given seminars/training in SharePoint and has also done consultancy work for several of the biggest SharePoint implementations in Europe, Africa, and South America. His articles can be found in many of the leading trade magazines in English, Dutch, German, and Spanish. He is the webmaster of `http://www.gavd.net`, the main Spanish language site dedicated to SharePoint. Gustavo is author and co-author of ten books about SharePoint, and founder and editor of `CompartiMOSS` an specialized magazine about Microsoft technologies.

Foreword

Writing a new book is always a big experience and an adventure. And, in this case, it is not an adventure that will be finished when the book is ready: It will be a continuous endeavor for many years. Following the main idea and guidelines of Güitaca Publishers, this is a book that will reinvent itself again and again, following the Microsoft Office 365 constant evolution.

After working with Microsoft SharePoint Server since its introduction, it was somehow difficult to switch to SharePoint Online at first. I came from SharePoint implementations where everything was customized, where everything the customer wanted was possible to be done. And, suddenly, I had to work wearing a very tight corset that hindered my freedom as an architect and developer; furthermore, the product was no more "mine", but owned by Microsoft. When the online version started evolving in a freer direction, I found back the joy that SharePoint Server always gave me. And, in the latest years, exploring and implementing not only all the new products in the Office suite but also the countless possibilities of interaction with Microsoft Azure, my professional life has not only been enriched, but has also become more exhilarating (and complicated).

This book would not have been possible without the support of the complete team of Güitaca Publishers. The incessant encouragement of Vicky (the editor in chief) has been heartwarming, in a way only she can do it. And the backing of Alcira (in charge of the linguistics team), helping me through the labyrinths of a language that is not mine, has been invaluable. I also need to thank Vadim Gremyachev (colleague MVP and developer, https://blog.vgrem.com), and John Gillard (from ABCpdf, http://www.websupergoo.com) for their generosity helping me with software to support my work.

I hope you will enjoy this book and find it helpful following your path through Office 365.

With kindest regards,

Gustavo Velez

Book structure

"Office 365: the best recipes for developers" is a book aimed at coders. It explains how to work programmatically with Microsoft Office 365.

Office 365 is the collaboration and information sharing platform of Microsoft. It offers a few servers (Exchange, SharePoint), editing and authoring tools (Outlook, Word, Excel, PowerPoint, Visio), and a myriad of other applications to help businesses in creating, managing and organizing information.

Whom is this book for?

This is a book made by and targeted to developers. We assume the readers do know how the Office applications work: You will not find functional descriptions or instructions for use. The book is also for developers that know the programming tools and technologies used by Microsoft and Office 365: Visual Studio, Visual Studio Code, CSharp, PowerShell, JavaScript, etc.

To develop with Office, if you do not have a license, you can join the Office Developer Program to get one and do the necessary development with the Office 365 suite of programs:

https://developer.microsoft.com/en-us/office/dev-program.

The subscription is valid for 90 days and can be renewed for another 90 days for as long as you are using it for development activity.

Books by subscription

This is one in the series of books published by Güitaca under the mantra "Books by Subscription". Because modern technologies, and specially cloud software, evolve in a very fast tempo, the only way to maintain our readers actualized is publishing books that progress at the same speed as the technology.

You can buy the book in three ways:

- **Book plus subscription** for 12 updates. You will get the latest book version, access to the monthly updates, access to the repositories with the source code, and support from the author.
- **Book only**. You will get the first version of the book. You will miss the updates, will have no access to the source code repositories nor any kind of support. This is the case if you buy the book from Amazon or any other place and not directly from Güitaca (see the following section if this is your case).
- **Subscription only**. If you bought the book only and want to update it to the latest version, or if your subscription has expired and you want to renew it.

If you have a subscription, every four to six weeks you will receive an update that includes:

- Changes, modifications, and additions made by Microsoft to the technologies explained in the book.
- New content. We are aware that the book, probably, will never comprise every aspect of Office 365. Also, we do not have the arrogance to say that the author knows everything about Office. But we can ensure that we are aiming to make the content as complete as we can.
- Additionally, you get support from the author. If you have any question about the book or the published code, just let us know (info@guitaca.com) and we will try to help you as soon as possible (if we can, of course).

After 12 updates, we will produce a new edition, consolidating the last updates, adding new content if necessary, and initializing a new cycle. If you bought the book with subscription, you do not need to pay for it again: Your book registration will comprise the new edition as well if the subscription is valid.

If you bought the book from Amazon or another provider

If you bought your book from Amazon or any other provider, you will only get the book, will have no updates subscription, no access to the source code and no support from the author.

You can update your book getting the complete experience by going to our site https://guitaca.com, and buying the Subscription only offering. You only need to send us the receipt you got from Amazon or other provider, and you will get the full and most recent book version, access to the following 12 updates, access to the source code repositories, and support from the author.

Your book registration

The book updates are available from our download site https://guitaca.com/Home/Login. When you buy the book from our site (or the Subscription deal if you bought your book from a provider other than Güitaca), you receive a License Key sent to you with the receipt or after our validation. With your email address and this license code, you can log in to the downloads site. If you lose the code, you can ask for a new one from the site (a confirmation email will be sent to your email address, and you need to validate it).

Data privacy statement:

We only ask for an email address for registration of your book's copy and make the downloads available for you. **We DON'T use it for any commercial purpose, and we will never monetize it in any way**. You will only receive a few emails from us:

- When you buy the book (or subscription) from our site, you receive one email with the acknowledgement of receipt and your License Key code.
- If you lose your password and request a new one, we will send one validation email first and then another one, after validation, with the new License Key to the registered address.
- By default, we will send you one email when a new update is ready for download. You can void it at any moment from the download site self (and you can change it again if you wish).
- If your subscription is coming to an end, we will send you one email with instructions on how to renew it.

About the source code

All the source code was developed using Visual Studio Enterprise and, in some cases, Visual Studio Code. We always use the last version of Visual Studio, with the latest patches installed. Each fragment of code (recipe) was tested to ensure the code is functioning correctly. Whenever possible, we also made unit-test classes (not available for download), so that we can automatically test the code if Microsoft changes something in the APIs.

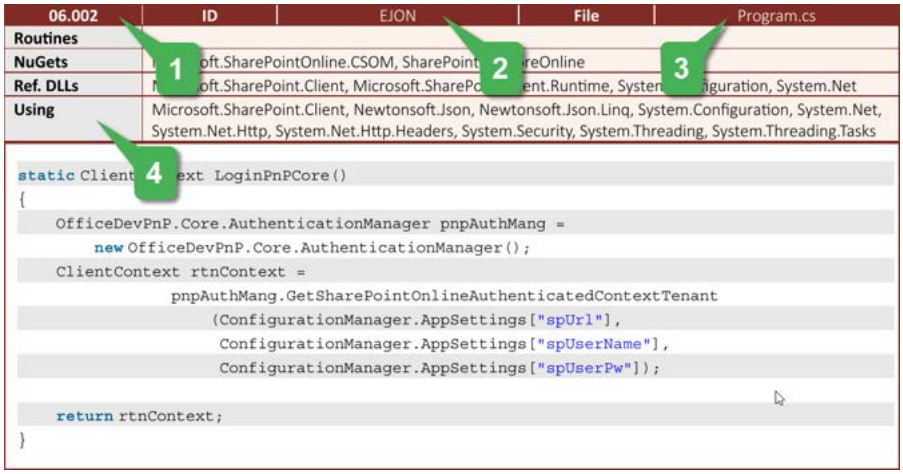
Although Visual Studio Enterprise was used for the code, you can also use Visual Studio Community, which can be downloaded for free from the Microsoft site:

https://visualstudio.microsoft.com/downloads/

Visual Studio Code is an Open Source and free code editor that can be downloaded from the same site, and is used for some parts of the book as well.

The code is simplified as much as possible to minimize its size: It does not include, for example, exception routines, and it should not be used directly for production applications.

All the source code can be downloaded from our source code repositories. To locate the solution used, each source fragment in the book has a header in the following form (if you bought the book only offered from Amazon or other provider, your copy will not have the information about the repo and code location):



Where:

1 is the chapter number and sequential number of the recipe in the chapter. If you need to send us a comment about any code fragment in the book, please refer to this number.

2 is the solution code identifier. This is the name of the solution that you will find in the

repository.

3 is the file name where the recipe can be found in the solution.

4 is the additional information about the routines, NuGets, dlls, etc., required to make the recipe work.

Amsterdam - Bogotá - Bilbao

Table of Content

End of TOC

01

Exchange Online

1. Exchange Online

Exchange Online is the hosted version for the messaging platform in Microsoft Office 365 that provides organizations with access to the full-featured version of the traditional Onprem Exchange Server. Microsoft Exchange Online is among the most mature of Microsoft's cloud offerings, being part of the Office cloud offering from the beginning, when Office 365 was called Business Productivity Online Suite (BPOS). That also means that Exchange is almost fully developed and there has not been any new main functionality added for years.
The same can be said about the development possibilities of Exchange: The current API, the Exchange Web Services (EWS), hasn't changed in many years, although it is becoming replaced by the Microsoft Graph API, which is progressively introducing a new REST interface for the server.

1.1. Introduction to developing for Exchange

Exchange Online offers three main possibilities to be accessed programmatically: Exchange Web Services API (EWS), a dedicated set of PowerShell cmdlets (called Exchange Online PowerShell), and, since the introduction of Microsoft Graph, the possibility to approach Exchange Online using REST APIs has been open. Because Graph is an ongoing project by Microsoft, the functionality it offers is not (yet) as complete as the possibilities given by EWS, but Graph is officially replacing EWS. The Exchange Online PowerShell is mainly used for configuration, monitoring, maintenance, and to manage Exchange from the command line.

EWS is becoming deprecated: Microsoft has announced (https://developer.microsoft.com/en-us/graph/blogs/upcoming-changes-to-exchange-web-services-ews-api-for-office-365/) that starting on July 19th, 2018 "...Exchange Web Services (EWS) will no longer receive feature updates. While the service will continue to receive security updates and certain non-security updates, product design and features will remain unchanged. This change also applies to the EWS SDKs for Java and .NET as well."

EWS can be used in precisely the same way for Exchange Online and Exchange Onprem, the only difference is given by the disparities in functionality in the two systems (they are very similar in any way), and the login method. EWS can be used in different ways: As a managed API by any development language (making SOAP, Simple Object Access Protocol, calls under the hood), as a set of SOAP Web Services, and from PowerShell. Because SOAP has been replaced by REST in the enterprise and is, in fact, not used anymore, its programmatic use will be not discussed in this book.

Regarding PowerShell, there is a set of cmdlets dedicated to Exchange Online Protection (EOP), but they are only used in standalone EOP organizations (for example, to protect an OnPremises Exchange environment). If you are working with an Office 365 subscription that includes EOP (E3, E5, etc.), you don't use Exchange Online Protection PowerShell; the same features are available in the standard Exchange Online PowerShell modules.

1.2. Login in Exchange

As it happens with other servers from Office 365, it is necessary to log in to the system to be authenticated, before any program can start interacting with the data.

1.2.1. Login from a managed language (CSharp)

For the CSharp examples in this chapter, the values to log in to Exchange (email address, password, application ID, and tenant ID, depending on the authorization method) are saved in an external file called

exCs.values.config that is used by the appSettings section in the App.Config Visual Studio Solution file. The App.Config file for the Visual Studio Solution contains the following section inside the <configuration> tag:

```
<appSettings file="c:\Temporary\exCs.values.config">
 <add key="exUserName" value="" />
 <add key="exUserPw" value="" />
 <add key="exAppId" value="" />
 <add key="exTenantId" value="" />
</appSettings>
```

The first line points to an external file that contains the values to be used by the App.Config file; it has the following form:

```
<appSettings >
 <add key="exUserName" value="user@domain.onmicrosoft.com" />
 <add key="exUserPw" value="VerySecurePw" />
 <add key="exAppId" value="SomeGuid" />
 <add key="exTenantId" value="SomeGuid" />
 </appSettings>
```

The values can be called by their name in the appSettings file, as follows:

```
string myExUser = ConfigurationManager.AppSettings["exUserName"];
```

1.2.2. Login from PowerShell

EWS can be used by PowerShell to reach programmatically the Exchange information. The Microsoft.Exchange.WebServices.dll needed to work with EWS must be installed locally.

The EWS DLLs can be downloaded from `https://www.microsoft.com/en-us/download/details.aspx?id=42951`. Download the EwsManagedApi.msi file from that site and install it locally. The DLLs will be installed in the local directory `C:\Program Files\Microsoft\Exchange\Web Services\2.2\`.

To make the EWS DLLs available for PowerShell, they must be loaded at the beginning of the script with

the following statement (which must be in one continuous line):

```
Add-Type -Path "C:\Program Files\Microsoft\Exchange\Web Services\2.2
\Microsoft.Exchange.WebServices.dll"
```

The values to log in to Exchange (email address, password, application ID, and tenant ID, depending on the authorization method) are saved in an external file called exPs.values.config (in XML format) that is loaded by PowerShell at runtime. The external config file is called at the beginning of the script, as follows:

```
[xml]$configFile = get-content "C:\Projects\exPs.values.config"
```

And the file exPs.values.config contains the values to be used in the script; it has the following form:

```
<appSettings>
 <exUserName>user@domain.onmicrosoft.com</exUserName>
 <exUserPw>VerySecurePw</exUserPw>
 <exAppId>SomeGuid</exAppId>
 <exTenantId>SomeGuid</exTenantId>
</appSettings>
```

The values can be called by their name in the appSettings file, as follows:

```
$myUser = $configFile.appsettings.exUserPw
```

Note: If PowerShell is not allowing to run scripts, change the execution policy using the command:

```
Set-ExecutionPolicy -ExecutionPolicy [Unrestricted]/[RemoteSigned]/[Default]
```

1.2.3. Login with user credentials for EWS (Basic Authentication) and CSharp

Microsoft recommends not to use basic authentication (using username/password) anymore, although Exchange Online still accepts this type of authentication. Nevertheless, basic authentication can be a good option, to avoid extensive setup tasks and repetitive logins, for simple test or demonstration applications.

EWS Basic Authentication is becoming fully decommissioned: Microsoft has announced (https://developer.microsoft.com/en-us/graph/blogs/upcoming-changes-to-exchange-web-services-ews-api-for-office-365/) that on October 13th, 2020, it "...will stop supporting and fully decommission the Basic Authentication for EWS to access Exchange Online". This means that new or existing apps will not be able to use Basic Authentication when connecting to Exchange using EWS.

The function (ConnectBA) in the following recipe uses the email address and password of one user to get authorized in Exchange. The AutodiscoverUrl method determines the best endpoint for a given user (the endpoint that is closest to the user's Mailbox server); this method can be called using only the username parameter, but Exchange Online rejects the request as unsafe. Therefore, the RedirectionUrlValidationCallback routine, which is considered valid if it uses HTTPS, must be used in conjunction with the authentication call. Instantiating the ExchangeService with an empty constructor will create an instance that is bound to the latest known version of Exchange. The TraceEnabled and TraceFlag properties can be activated to get information from Exchange about the login process (for debugging purposes), and the Url method of the service instance gives back the address used by Exchange Online.

01.001	ID	File
Routines		
NuGets	Microsoft.Exchange.WebServices, Microsoft.Identity.Client	
Ref. DLLs	Microsoft.Exchange.WebServices, Microsoft.Exchange.WebServices.Auth, Microsoft.Identity.Client	
Using	Microsoft.Exchange.WebServices.Data, Microsoft.Identity.Client	

```
static ExchangeService ConnectBA(string userEmail, string userPW)
{
    ExchangeService exService = new ExchangeService
    {
        Credentials = new WebCredentials(userEmail, userPW)
    };

    //exService.TraceEnabled = true;
    //exService.TraceFlags = TraceFlags.All;

    exService.AutodiscoverUrl(userEmail, RedirectionUrlValidationCallback);
    //Console.WriteLine(exService.Url);

    return exService;
}
```

```
static bool RedirectionUrlValidationCallback(string redirectionUrl)
{
    bool validationResult = false;

    Uri redirectionUri = new Uri(redirectionUrl);
    if (redirectionUri.Scheme == "https")
    {
        validationResult = true;
    }

    return validationResult;
}
```

The authorization method can be called from any other routine as follows:

01.002	ID		File
Routines			
NuGets	Microsoft.Exchange.WebServices, Microsoft.Identity.Client		
Ref. DLLs	Microsoft.Exchange.WebServices, Microsoft.Exchange.WebServices.Auth, Microsoft.Identity.Client		
Using	Microsoft.Exchange.WebServices.Data, Microsoft.Identity.Client		

```
static void Main(string[] args)
{
    ExchangeService myExService = ConnectBA(
                        ConfigurationManager.AppSettings["exUserName"],
                        ConfigurationManager.AppSettings["exUserPw"]);

    CallEWSTest(myExService);
}
```

1.2.4. Login with oAuth for EWS and CSharp

Although the fact that oAuth relies on a third-party authentication provider, that the standard is more difficult to implement than basic authentication, and that oAuth requires another layer of integration (the application will need both, the authentication provider and the Exchange server), Microsoft recommends using oAuth instead of basic authentication because of the advantage in security.

Since Office 365 uses Azure Active Directory (AAD) as authentication provider, any application that wants to use Office Exchange EWS oAuth authentication must have an application ID issued by AAD. The following steps indicate how to register one application as a public client with Azure Active Directory.

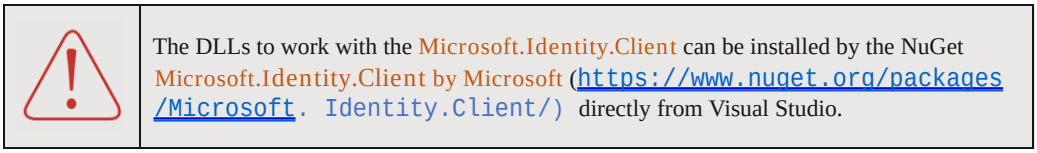
1 - Using a browser, navigate to the main administration page of Office 365 (https://admin.microsoft.com or through https://portal.office.com), log in with an administrator account, and open the Azure Active Directory Admin Center.

2 - Click on Azure Active Directory in the menu on the left side, and then on App registrations (Manage section). Use the New registration button.

3 - Assign a name to the registration, select Accounts in this organizational directory only in the Supported account types section, and select the Public client/native (mobile & desktop) option in the Redirect Uri section. Write the value urn:ietf:wg:oauth:2.0:oob on the textbox at the side of the Redirect Uri section. Use the Register button to save the registration.

4 - The registration is complete. Copy the values given in Application (client) ID and Directory (tenant) ID to use it in the source code of the application to be developed.

The Visual Studio solution to use the authentication from oAuth (a console applications in this chapter) requires a using directive to Microsoft.Identity.Client.



The DLLs to work with the Microsoft.Identity.Client can be installed by the NuGet Microsoft.Identity.Client by Microsoft (https://www.nuget.org/packages/Microsoft. Identity.Client/) directly from Visual Studio.

The function ConnectOA in the following recipe uses the Azure AD registration client and tenant ID to get authorized in Exchange.

01.003	ID	File
Routines		
NuGets	Microsoft.Exchange.WebServices, Microsoft.Identity.Client	
Ref. DLLs	Microsoft.Exchange.WebServices, Microsoft.Exchange.WebServices.Auth, Microsoft.Identity.Client	
Using	Microsoft.Exchange.WebServices.Data, Microsoft.Identity.Client	

```
static async System.Threading.Tasks.Task<ExchangeService> ConnectOA(
                                                    string AppId, string TenId)
{
    ExchangeService exService = new ExchangeService();

    PublicClientApplicationOptions pcaOptions = new PublicClientApplicationOptions
    {
        ClientId = AppId,
        TenantId = TenId
    };

    IPublicClientApplication pcaBuilder = PublicClientApplicationBuilder
        .CreateWithApplicationOptions(pcaOptions).Build();

    string[] exScope = new string[] {
                    "https://outlook.office.com/EWS.AccessAsUser.All" };

    AuthenticationResult authToken = await
                      pcaBuilder.AcquireTokenInteractive(exScope).ExecuteAsync();
```

```
    exService.Url = new Uri("https://outlook.office365.com/EWS/Exchange.asmx");
    exService.Credentials = new OAuthCredentials(authToken.AccessToken);

    return await System.Threading.Tasks.Task.FromResult(exService);
}
```

The first time that the application runs, a standard login window will appear requiring the account data of the user that made the registration. After login, the window will ask for permissions (Access your mailboxes, Maintain access to data you have given it access to and View your basic profile). Subsequently, the application will ask only for the user login, not for the permissions.

Because the connection routine is asynchronous, use the following code to call it.

01.004	ID	File
Routines		
NuGets	Microsoft.Exchange.WebServices, Microsoft.Identity.Client	
Ref. DLLs	Microsoft.Exchange.WebServices, Microsoft.Exchange.WebServices.Auth, Microsoft.Identity.Client	
Using	Microsoft.Exchange.WebServices.Data, Microsoft.Identity.Client	

```
static void Main(string[] args)
{
    ExchangeService myExService = ConnectOA(
                        ConfigurationManager.AppSettings["exAppId"],
                        ConfigurationManager.AppSettings["exTenantId"]).
                                                GetAwaiter().GetResult();

    CallEWSTest(myExService);
}
```

The use of oAuth to get access to Exchange is faster than the use of base authentication, especially because it is not necessary to use the auto-discovery method.

1.2.5. Login for EWS with PowerShell and Basic Authentication

The ConnectPsEwsBA routine in the next recipe takes care of login in Exchange using Basic Authentication with PowerShell.

01.005	ID	File
Routines		
PS Modules	GenericOauthEWS.ps1	
Other Modules	Microsoft.Exchange.WebServices.dll	

```
Function ConnectPsEwsBA()
{
    $ExService = New-Object Microsoft.Exchange.WebServices.Data.ExchangeService
    $ExService.Credentials = New-Object
Microsoft.Exchange.WebServices.Data.WebCredentials(`
        $configFile.appsettings.exUserName, $configFile.appsettings.exUserPw)
    $ExService.Url = new-object Uri("https://outlook.office365.com/EWS/Exchange.asmx");
    #$ExService.TraceEnabled = $true
    #$ExService.TraceFlags = [Microsoft.Exchange.WebServices.Data.TraceFlags]::All
    $ExService.AutodiscoverUrl($configFile.appsettings.exUserName, {$true})

    return $ExService
}
```

See section 1.2.2. to get details about login with PowerShell. The TraceEnabled and TraceFlag properties can be activated at any moment to get information regarding the internal working of Exchange when logging in.

To call the function, use code similar to the next example.

01.006	ID	File
Routines		
PS Modules	GenericOauthEWS.ps1	
Other Modules	Microsoft.Exchange.WebServices.dll	

```
##==> EWS Basic Authorization
Add-Type -Path "C:\Program Files\Microsoft\Exchange\Web Services\2.2
\Microsoft.Exchange.WebServices.dll"
$ExService = ConnectPsEwsBA

CallEWSTest $ExService #Calling any function
```

1.2.6. Login for EWS with PowerShell and oAuth

Using oAuth from PowerShell is not a trivial or easy endeavor. But because Basic Authentication is being closed for Exchange, it will be obligatory to use in some years.

For this book, we use the GenericOauthEWS.ps1 login routine developed by Glen Scales (glenscales@yahoo.com, https://gsexdev.blogspot.com/), that can be downloaded from his GitHub repository https://github.com/gscales/Powershell-Scripts/blob/master/GenericOauthEWS.ps1. The module is fully described in the article https://gsexdev.blogspot.com/2018/08/dependency-free-generic-ews-oauth.html.

The EWS DLLs required to work with oAuth and Exchange can be downloaded from https://www.microsoft.com/en-us/download/details.aspx?id=42951. Download the EwsManagedApi.msi file from that site and install it locally. The DLLs will be installed in the local directory C:\Program Files\Microsoft\Exchange\Web Services\2.2\.

To use the module, first, load it in the script and then call the Connect-Exchange method. Use the return value to get any access to the required information.

01.007	ID		File
Routines			
PS Modules	GenericOauthEWS.ps1		
Other Modules	Microsoft.Exchange.WebServices.dll		

```
##==> EWS oAuth Authorization
Import-Module .\GenericOauthEWS.ps1 -Force
#Test-EWSConnection -MailboxName $configFile.appsettings.exUserName
$ExService = Connect-Exchange `
                $configFile.appsettings.exUserName "" $configFile.appsettings.exAppId

CallEWSTest $ExService #Calling any function
```

The module has a Test-EWSConnection method that can be used to check the connection with Exchange and get some information about the account.

1.2.7. Login using the Exchange Online PowerShell

To work with Exchange Online PowerShell, use Windows PowerShell on the local computer to create a remote PowerShell session with Exchange Online, providing the Office 365 credentials, the required connection settings, and then import the Exchange Online cmdlets into the local Windows PowerShell session.

The following function automates the complete process.

01.008	ID		File
Routines			
PS Modules	GenericOauthEWS.ps1		
Other Modules	Microsoft.Exchange.WebServices.dll		

```
Function ConnectPsOnlBA()
{
    [SecureString]$securePW = ConvertTo-SecureString -String `
                $configFile.appsettings.exUserPw -AsPlainText -Force
    $myCredentials = New-Object System.Management.Automation.PSCredential -ArgumentList `
```

```
                $configFile.appsettings.exUserName, $securePW
    $mySession = New-PSSession -ConfigurationName Microsoft.Exchange -ConnectionUri `
                https://outlook.office365.com/powershell-liveid/ -Authentication Basic `
                -AllowRedirection -Credential $myCredentials
    Import-PSSession $mySession -AllowClobber
}
```

To use the remote session, utilize the function as follows.

01.009	ID		File
Routines			
PS Modules	GenericOauthEWS.ps1		
Other Modules	Microsoft.Exchange.WebServices.dll		

```
##==> Exchange Online PowerShell Basic Authorization
ConnectPsOnlBA

Get-Mailbox #Calling any cmdlet

$currentSession = Get-PSSession
Remove-PSSession -Session $currentSession
```

The remote session is killed at the end of the code for security concerns.

Microsoft has announced that Remote PowerShell for Exchange will be closed on October 13th, 2020 (https://techcommunity.microsoft.com/t5/blogs/blogarticleprintpage/blog-id/Exchange/article-id/27095). Microsoft recommends using the multi-factor authentication PowerShell module or the PowerShell within Azure Cloud Shell to use PowerShell with Exchange, as the article mentions.

1.3. Programming Exchange with EWS and CSharp

To work with EWS in Visual Studio, the development computer must have the Microsoft.Exchange.WebServices and Microsoft.Exchange.WebServices.Auth DLLs installed.

The EWS DLLs can be installed by the NuGet Microsoft.Exchange.WebServices by Microsoft (https://www.nuget.org/packages/Microsoft.Exchange.WebServices/) directly from Visual Studio.

08/ *SharePoint Online Site Collections and Webs*

8. SharePoint Online Site Collections and Webs

For SharePoint Online, the Site Collection is the biggest container for maintaining information. A Site Collection, as its name indicates, contains at least one site (the root Web), but can host a complete structure of subsites (the SharePoint Webs).
A Site Collection offers site users unified navigation, branding, security, and search tools as a cohesive website experience.
The following recipes should mostly work as well for the modern SharePoint user experience as for the classic user experience. The cases where they use different methods will also be explained in the text.

8.1. Introduction

This chapter shows the basic CRUD (Create, Read, Update, Delete) recipes to work with Site Collections and Webs, using the SharePoint Client Object Model (CSOM), PnP, and PowerShell. Extra information about security, configuration, etc., is also included.

All recipes use the login methods presented in Chapter 06, and the routine's code is not repeated in this chapter. Please review Chapter 06 for login code and configuration instruction.

All the recipes have been developed for, and tested with, Modern SharePoint Site Collections. Because almost all the APIs were developed originally to work with the Classic SharePoint user experience, the recipes will work generally without problems as well for the old experience of Site Collections and Webs.

8.2. Operations for modern Site Collections with the Client Side Object Model (CSharp)

The SharePoint Client Side Object Model (CSOM) is designed to work with SharePoint elements from the Site Collection level to the lowest architecture elements (Items and Documents). For this reason, the CSOM is not able to work at the tenant level, and it has no methods to, for example, create or enumerate Site Collections. To work with the highest rank of elements in the SharePoint hierarchy, it is necessary to use the Microsoft.Online.SharePoint namespace. The necessary assemblies to do that are also installed, together with the Microsoft.SharePoint.Client assemblies, when the NuGet Microsoft.SharePointOnline.CSOM is added to the Visual Studio Solution.

The following recipes will use both namespaces indiscriminately. When operations at tenant level are used, it is necessary to reference the administration Site of SharePoint Online, and use a SharePoint administrator account. The login routines are the same (as indicated in Chapter 06) for employing the administration site (`http://domain-admin.sharepoint.com`) or a normal Site Collection (`http://domain.sharepoint.com/sites/sitecoll`); the only difference is the URL to use.

8.2.1. Creation of modern Site Collections - CSOM, CSharp

Only SharePoint administrator accounts can create Site Collections in SharePoint Online. There are two types of Site Collections: based on the modern SharePoint user experience and based on the classic experience. How to create modern team sites programmatically depends on whether it needs to be connected to an Exchange Group or not.

For non-group connected sites, a call to a CSOM method for creating sites, and passing in the template identifier STS#3 (for a Team Site) or SITEPAGEPUBLISHING#0 (for a Communication Site) will suffice. For classic Site Collections, use any of the other template identifiers.

08.001	ID	File
Routines	LoginCsom (see Ch06-s6.1)	
NuGets	Microsoft.SharePointOnline.CSOM	
Ref. DLLs		
Using	Microsoft.SharePoint.Client, Microsoft.Online.SharePoint.TenantAdministration, System.IO	

```
static void SpCsCsomCreateOneSiteCollection(ClientContext spAdminCtx)
{
    Tenant myTenant = new Tenant(spAdminCtx);
    string myUser = ConfigurationManager.AppSettings["spUserName"];
    SiteCreationProperties mySiteCreationProps = new SiteCreationProperties
    {
        Url = ConfigurationManager.AppSettings["spBaseUrl"] +
                                        "/sites/NewSiteCollectionModernCsCsom01",
        Title = "NewSiteCollectionModernCsCsom01",
        Owner = ConfigurationManager.AppSettings["spUserName"],
        Template = "STS#3",
        StorageMaximumLevel = 100,
        UserCodeMaximumLevel = 50
    };

    SpoOperation myOps = myTenant.CreateSite(mySiteCreationProps);
    spAdminCtx.Load(myOps, ic => ic.IsComplete);
    spAdminCtx.ExecuteQuery();

    while (myOps.IsComplete == false)
    {
        System.Threading.Thread.Sleep(5000);
        myOps.RefreshLoad();
        spAdminCtx.ExecuteQuery();
    }
}
```

For a Group connected modern site, create an Office 365 group first, and determine the name of the team site to connect to, as shown in the next routine.

08.002	ID	File
Routines	LoginCsom (see Ch06-s6.1)	
NuGets	Microsoft.SharePointOnline.CSOM	
Ref. DLLs		
Using	Microsoft.SharePoint.Client, Microsoft.Online.SharePoint.TenantAdministration, System.IO	

```
static void SpCsCsomCreateGroupForSite(ClientContext spAdminCtx)
{
    string[] myOwners = new string[] { "user@domain.onmicrosoft.com" };
    GroupCreationParams myGroupParams = new GroupCreationParams(spAdminCtx);
    myGroupParams.Owners = myOwners;
    //GroupCreationParams
```

```
    Tenant myTenant = new Tenant(spAdminCtx);
    myTenant.CreateGroupForSite(
        ConfigurationManager.AppSettings["spBaseUrl"] +
                                        "/sites/NewSiteCollectionModernCsCsom01",
        "GroupForNewSiteCollectionModernCsCsom01",
        "GroupForNewSiteCollAlias",
        true,
        myGroupParams);

    spAdminCtx.ExecuteQuery();
}
```

To find the identifiers for the different types of Site Collections, use the GetSPOTenantWebTemplates method, indicating the language location identifier.

08.003	ID	File
Routines	LoginCsom (see Ch06-s6.1)	
NuGets	Microsoft.SharePointOnline.CSOM	
Ref. DLLs		
Using	Microsoft.SharePoint.Client, Microsoft.Online.SharePoint.TenantAdministration, System.IO	

```
static void SpCsCsomFindWebTemplates(ClientContext spAdminCtx)
{
    Tenant myTenant = new Tenant(spAdminCtx);
    SPOTenantWebTemplateCollection myTemplates =
                            myTenant.GetSPOTenantWebTemplates(1033, 0);
    spAdminCtx.Load(myTemplates);
    spAdminCtx.ExecuteQuery();

    foreach (SPOTenantWebTemplate oneTemplate in myTemplates)
    {
        Console.WriteLine(oneTemplate.Name + " - " + oneTemplate.Title);
    }
}
```

8.2.2. Enumeration of Site Collections in the Tenant - CSOM, CSharp

There are no methods at the moment to enumerate modern Site Collections in SharePoint Online. To get the classic Site Collections in the tenant, use the GetSiteProperties method.

08.004	ID	File
Routines	LoginCsom (see Ch06-s6.1)	
NuGets	Microsoft.SharePointOnline.CSOM	
Ref. DLLs		
Using	Microsoft.SharePoint.Client, Microsoft.Online.SharePoint.TenantAdministration, System.IO	

```
static void SpCsCsomReadAllSiteCollections(ClientContext spAdminCtx)
{
    Tenant myTenant = new Tenant(spAdminCtx);
    myTenant.GetSiteProperties(0, true);

    SPOSitePropertiesEnumerable myProps = myTenant.GetSiteProperties(0, true);
    spAdminCtx.Load(myProps);
    spAdminCtx.ExecuteQuery();

    foreach (var oneSiteColl in myProps)
    {
        Console.WriteLine(oneSiteColl.Title + " - " + oneSiteColl.Url);
    }
}
```

8.2.3. Delete Site Collections from the Tenant - CSOM, CSharp

The RemoveSite method deletes a Site Collection from the tenant if it has no connection to an Exchange group.

08.005	ID	File
Routines	LoginCsom (see Ch06-s6.1)	
NuGets	Microsoft.SharePointOnline.CSOM	
Ref. DLLs		
Using	Microsoft.SharePoint.Client, Microsoft.Online.SharePoint.TenantAdministration, System.IO	

```
static void SpCsCsomRemoveSiteCollection(ClientContext spAdminCtx)
{
    Tenant myTenant = new Tenant(spAdminCtx);
    myTenant.RemoveSite(
        ConfigurationManager.AppSettings["spBaseUrl"] +
                                        "/sites/NewSiteCollectionModernCsCsom01");

    spAdminCtx.ExecuteQuery();
}
```

To recover a Site Collection that has been deleted to the Recycle Bin, use the RestoreDeletedSite method. Take into consideration that the Recycle Bin removes its information automatically after some time.

08.006	ID	File
Routines	LoginCsom (see Ch06-s6.1)	
NuGets	Microsoft.SharePointOnline.CSOM	
Ref. DLLs		
Using	Microsoft.SharePoint.Client, Microsoft.Online.SharePoint.TenantAdministration, System.IO	

```
static void SpCsCsomRestoreSiteCollection(ClientContext spAdminCtx)
{
    Tenant myTenant = new Tenant(spAdminCtx);
    myTenant.RestoreDeletedSite(
        ConfigurationManager.AppSettings["spBaseUrl"] +
                                        "/sites/NewSiteCollectionModernCsCsom01");

    spAdminCtx.ExecuteQuery();
}
```

A Site Collection can be also deleted from the Recycle Bin using the RemoveDeletedSite method. That could be necessary to create a new Site Collection with the same name as an already deleted Site Collection.

08.007	ID	File
Routines	LoginCsom (see Ch06-s6.1)	
NuGets	Microsoft.SharePointOnline.CSOM	
Ref. DLLs		
Using	Microsoft.SharePoint.Client, Microsoft.Online.SharePoint.TenantAdministration, System.IO	

```
static void SpCsCsomRemoveDeletedSiteCollection(ClientContext spAdminCtx)
{
    Tenant myTenant = new Tenant(spAdminCtx);
    myTenant.RemoveDeletedSite(
        ConfigurationManager.AppSettings["spBaseUrl"] +
                                        "/sites/NewSiteCollectionModernCsCsom01");

    spAdminCtx.ExecuteQuery();
}
```

8.2.4. Add users with rights to one Site Collection - CSOM, CSharp

The isSiteAdministrator parameter (last parameter) in the SetSiteAdmin method indicates if a newly added account to the security settings of the Site Collection is an administrator.

08.008	ID	File
Routines	LoginCsom (see Ch06-s6.1)	
NuGets	Microsoft.SharePointOnline.CSOM	
Ref. DLLs		
Using	Microsoft.SharePoint.Client, Microsoft.Online.SharePoint.TenantAdministration, System.IO	

```
static void SpCsCsomSetAdministratorSiteCollection(ClientContext spAdminCtx)
{
    Tenant myTenant = new Tenant(spAdminCtx);
    myTenant.SetSiteAdmin(
        ConfigurationManager.AppSettings["spBaseUrl"] +
                                        "/sites/NewSiteCollectionModernCsCsom01",
        "user@domain.onmicrosoft.com",
        true);

    spAdminCtx.ExecuteQuery();
}
```

8.2.5. Working with modern Hub Sites - CSOM, CSharp

A modern Hub Site Collection is a logical aggregator of Site Collections. Any modern Teams Site Collection can be elevated to Hub Site Collection.

08.009	ID	File
Routines	LoginCsom (see Ch06-s6.1)	
NuGets	Microsoft.SharePointOnline.CSOM	
Ref. DLLs		
Using	Microsoft.SharePoint.Client, Microsoft.Online.SharePoint.TenantAdministration, System.IO	

```
static void SpCsCsomRegisterAsHubSiteCollection(ClientContext spAdminCtx)
{
    Tenant myTenant = new Tenant(spAdminCtx);
    myTenant.RegisterHubSite(
        ConfigurationManager.AppSettings["spBaseUrl"] +
                                    "/sites/NewHubSiteCollCsCsom");

    spAdminCtx.ExecuteQuery();
}
```

In a similar way, a Site Collection can be demoted from Hub Site Collection back to normal modern Site Collection.

08.010	ID	File
Routines	LoginCsom (see Ch06-s6.1)	
NuGets	Microsoft.SharePointOnline.CSOM	
Ref. DLLs		
Using	Microsoft.SharePoint.Client, Microsoft.Online.SharePoint.TenantAdministration, System.IO	

```
static void SpCsCsomUnregisterAsHubSiteCollection(ClientContext spAdminCtx)
{
    Tenant myTenant = new Tenant(spAdminCtx);
    myTenant.UnregisterHubSite(
        ConfigurationManager.AppSettings["spBaseUrl"] +
                                    "/sites/NewHubSiteCollCsCsom");

    spAdminCtx.ExecuteQuery();
}
```

The GetHubSitePropertiesByUrl method gets the current information configured for a Hub Site Collection. There is also a GetHubSitePropertiesById to recover the Hub information given its identifier.

08.011	ID	File
Routines	LoginCsom (see Ch06-s6.1)	
NuGets	Microsoft.SharePointOnline.CSOM	
Ref. DLLs		
Using	Microsoft.SharePoint.Client, Microsoft.Online.SharePoint.TenantAdministration, System.IO	

```
static void SpCsCsomGetHubSiteCollectionProperties(ClientContext spAdminCtx)
{
    Tenant myTenant = new Tenant(spAdminCtx);
    HubSiteProperties myProps = myTenant.GetHubSitePropertiesByUrl(
        ConfigurationManager.AppSettings["spBaseUrl"] +
                                    "/sites/NewHubSiteCollCsCsom");

    spAdminCtx.Load(myProps);
    spAdminCtx.ExecuteQuery();

    Console.WriteLine(myProps.Title);
}
```

And the same method can be used to update the metadata of the Hub.

08.012	ID	File
Routines	LoginCsom (see Ch06-s6.1)	
NuGets	Microsoft.SharePointOnline.CSOM	
Ref. DLLs		
Using	Microsoft.SharePoint.Client, Microsoft.Online.SharePoint.TenantAdministration, System.IO	

```
static void SpCsCsomUpdateHubSiteCollectionProperties(ClientContext spAdminCtx)
{
    Tenant myTenant = new Tenant(spAdminCtx);
    HubSiteProperties myProps = myTenant.GetHubSitePropertiesByUrl(
        ConfigurationManager.AppSettings["spBaseUrl"] +
                                    "/sites/NewHubSiteCollCsCsom");

    spAdminCtx.Load(myProps);
    spAdminCtx.ExecuteQuery();

    myProps.Title = myProps.Title + "_Updated";
    myProps.Update();

    spAdminCtx.Load(myProps);
    spAdminCtx.ExecuteQuery();

    Console.WriteLine(myProps.Title);
}
```

A normal classic Teams Site Collection can be added to the collection of Site Collections managed by a Hub Site Collection.

08.013	ID	File
Routines	LoginCsom (see Ch06-s6.1)	
NuGets	Microsoft.SharePointOnline.CSOM	
Ref. DLLs		
Using	Microsoft.SharePoint.Client, Microsoft.Online.SharePoint.TenantAdministration, System.IO	

```
static void SpCsCsomAddSiteToHubSiteCollection(ClientContext spAdminCtx)
{
    Tenant myTenant = new Tenant(spAdminCtx);
    myTenant.ConnectSiteToHubSite(
        ConfigurationManager.AppSettings["spBaseUrl"] +
                                    "/sites/NewSiteForHub",
    ConfigurationManager.AppSettings["spBaseUrl"] +
                                    "/sites/NewHubSiteCollCsCsom");
    spAdminCtx.ExecuteQuery();
}
```

Any Site Collection that is in the collection of sites managed by a Hub can also be removed from the Hub.

08.014	ID		File	
Routines	LoginCsom (see Ch06-s6.1)			
NuGets	Microsoft.SharePointOnline.CSOM			
Ref. DLLs				
Using	Microsoft.SharePoint.Client, Microsoft.Online.SharePoint.TenantAdministration, System.IO			

```
static void SpCsCsomremoveSiteFromHubSiteCollection(ClientContext spAdminCtx)
{
    Tenant myTenant = new Tenant(spAdminCtx);
    myTenant.DisconnectSiteFromHubSite(
        ConfigurationManager.AppSettings["spBaseUrl"] +
                                    "/sites/NewSiteForHub");
    spAdminCtx.ExecuteQuery();
}
```

8.3. Operations for Webs in a Site Collection with the Client Side Object Model (CSharp)

8.3.1. Create Web Sites in a Site Collection - CSOM, CSharp

Use the WebCreationInformation method to configure the parameters of a new Web Site and add it to the Webs collection of the Site Collection. This recipe can create modern and classic experience Webs.

08.015	ID		File	
Routines	LoginCsom (see Ch06-s6.1)			
NuGets	Microsoft.SharePointOnline.CSOM			
Ref. DLLs				
Using	Microsoft.SharePoint.Client, Microsoft.Online.SharePoint.TenantAdministration, System.IO			

```
static void SpCsCsomCreateOneWebInSiteCollection(ClientContext spCtx)
{
    Site mySite = spCtx.Site;

    WebCreationInformation myWebCreationInfo = new WebCreationInformation
    {
        Url = "NewWebSiteModernCsCsom",
        Title = "NewWebSiteModernCsCsom",
        Description = "NewWebSiteModernCsCsom Description",
        UseSamePermissionsAsParentSite = true,
        WebTemplate = "STS#3",
        Language = 1033
    };
```

```
    Web myWeb = mySite.RootWeb.Webs.Add(myWebCreationInfo);
    spCtx.ExecuteQuery();
}
```

8.3.2. Find the Webs of a Site Collection - CSOM, CSharp

To enumerate all the Webs in a Site Collection, recall first the Site object, and then loop through each Web in the Webs collection.

08.016	ID	File
Routines	LoginCsom (see Ch06-s6.1)	
NuGets	Microsoft.SharePointOnline.CSOM	
Ref. DLLs		
Using	Microsoft.SharePoint.Client, Microsoft.Online.SharePoint.TenantAdministration, System.IO	

```
static void SpCsCsomGetWebsInSiteCollection(ClientContext spCtx)
{
    Site mySite = spCtx.Site;

    WebCollection myWebs = mySite.RootWeb.Webs;
    spCtx.Load(myWebs);
    spCtx.ExecuteQuery();

    foreach (Web oneWeb in myWebs)
    {
        Console.WriteLine(oneWeb.Title + " - " + oneWeb.Url + " - " + oneWeb.Id);
    }
}
```

To find only one of the Webs in the Site Collection, create a context using the URL of the Web directly. Then, all its properties can be read.

08.017	ID	File
Routines	LoginCsom (see Ch06-s6.1)	
NuGets	Microsoft.SharePointOnline.CSOM	
Ref. DLLs		
Using	Microsoft.SharePoint.Client, Microsoft.Online.SharePoint.TenantAdministration, System.IO	

```
static void SpCsCsomGetOneWebInSiteCollection()
{
    string myWebFullUrl = ConfigurationManager.AppSettings["spUrl"] +
                                                    "/NewWebSiteModernCsCsom";
    ClientContext spCtx = LoginCsom(myWebFullUrl);
```

14

Microsoft Teams

14. Microsoft Teams

Microsoft Teams is the group collaboration application in the Office 365 suite. It helps teams to work together from one place, integrating conversations, files, notes, and multiple other internal and external tools. Technically speaking, Microsoft Teams is a combination of Office 365 Exchange Groups (email, calendar, meetings), SharePoint Online (Lists, Libraries, Sites, OneDrive), and Skype for Business (chat, calls, video). Additionally, it is an open based system that allows integrating external, commercial and customized applications in the same user interface.

14.1. Introduction

Microsoft announced Teams in November 2016, and launched the service worldwide on 14 March 2017. Since then, the development of the application has been stormy, adding new functionality almost every week. At the beginning, Teams was no more than the combination of some functionality of Exchange (Groups) and SharePoint (Libraries) in one user interface, and certain experimental extensibility options. But because Microsoft sees Teams as a key component in its strategy for Office 365, the development of new functionality and interoperability has been very fast. In May 2017, Microsoft announced Microsoft Teams was replacing Microsoft Classroom in Office 365 Education; in September 2017, it was made known that it will replace Skype for Business, and in 2018, that StaffHub will be retired and its functionality moved to Teams.

While Teams was engulfing other products, its ability to interact with the external world was also improving. Initially, it was only possible to add bots and a couple of connectors with external applications, but now, it is interoperable with hundreds of applications, and developers can create new connectors, message extensions, Webhooks, and SharePoint Framework components. Also, the support for Microsoft Graph (using REST services) is getting better and, in April 2019, the general availability of the Microsoft Teams PowerShell module was announced.

Although Teams is available for Office 365 Business Essentials, Business Premium, and Enterprise E1, E3, and E5 plans, if you need a test and development Teams instance, it is possible to create a free tenant with some limitations, but fully functional, from https://products.office.com/en-us/microsoft-teams/free.

There are several ways to extend the functionality of Teams:

- Tabs that provide a full-screen web experience, embedded in the main presentation zone of the Teams user interface.
- Bots that interact with members of a conversation through chat, and can respond to events.
- Webhooks and Connectors that enable external services to send and receive messages.
- Messaging extensions that allow users to interact with Web services through buttons and forms from the Teams client user interface.
- SharePoint Framework (SPFx) components that are created as SharePoint Client WebParts.

Teams is not a hosting service: The customizations added to Teams are always hosted externally. The package to add the functionality to Teams contains a manifest with metadata about the app (name, icons, etc.), and pointers to the web services of the app. Also, take into consideration that any functionality exposed in a Microsoft Teams app is publicly available over the internet. If the app provides access to confidential or protected information, the app self should take care of authentication and authorization.

14.2. Teams configuration for developing

Teams can be activated for all users from the Central Administration of Office 365. Also, the access can be configured by the user if necessary.

For developing, sideloading (installation of applications without using Microsoft's application-distribution method) must be activated at three levels before it can be used:

- From the Teams Admin Center, open the Terms apps section, click on Setup policies, and open the Global (Org-wide default) policy. Flip the Upload custom apps button to On and save the configuration.
- From the Teams Admin Center, open the Terms apps section and use the Org-wide app settings button. Move the Custom apps selector to On.
- Each Team has the option Allow members to upload custom apps in the Manage Team - Settings - Member permissions window that should be activated.

14.3. Developing for Teams and development tools

A customization for Teams consists of a web application of a series of JavaScript files that must be hosted outside Teams, and a manifest that ensures the liaison between the external application and Teams. In fact, any web application could, in theory, be connected to Teams.

The main development tool to create new functionality for Teams is Visual Studio, even though Visual Studio Code can eventually be used as well. Microsoft has made some tools available to facilitate the development of customizations for Teams. Additionally, other third-party tools can help, especially for debugging.

14.3.1. Location of the Teams objects

The components available in the Teams client are physically located in SharePoint, Exchange, and Skype:

Chat is formally linked to the Skype server. Skype is being replaced by Teams, and its complete functionality will be available in Teams.

Teams is physically one Site Collections in SharePoint, plus one Group in Exchange. They can be reached programmatically using the APIs for Teams, SharePoint, and Exchange.

Each Team has different components:

- Channels. There are two types of Channels:
  - Standard (Public) Channels that use the base SharePoint Site Collection and Exchange Group created for the Team.
  - Private Channels that use the same Exchange Group as the Team, and a separated SharePoint Site Collection. This Site Collection is not visible from the SharePoint Central Administration page, but it can be reached programmatically as any other SharePoint Site Collection.

Each Channel is formed of three default components:

- Conversations, that are saved in the Exchange Group. The Group API doesn't allow access to the Conversations.
- Files, that are saved in the Documents Library of the SharePoint root site in the Site Collection.

Each Channel has one folder in the Library to save its files. Full programming access granted using the SharePoint APIs.

- Wikies, saved as .mht files (one for each Wiki) in the Teams Wiki Data Library of SharePoint. The .mht files are MIME HTML formatted files which save HTML, images and other linked resources into a single file. The SharePoint APIs allow access to these files.

Calendar in Teams is the calendar of the user, saved in Exchange. It is reachable programmatically using the Exchange APIs.

14.3.2. Teams App Studio

The Teams App Studio is a tool to help you build apps for Teams. It facilitates to start developing or integrating own service, streamlines the creation of the manifest for the apps, and provides other tools like a Card Editor and a React control library.

Teams App Studio is also an app which can be found in the Teams store. Click on the Apps button in the Teams client and search for App Studio in the store. After installing the app, it will be reachable from the ellipse button (...) on the left side menu of the user interface.

Microsoft has announced that the React control library in App Studio will be deprecated in the future. It is recommended to use the Fluent-UI react controls from `https://microsoft.github.io/fluent-ui-react/`.

14.3.3. Teams Developer Preview

The Teams Developer Preview is a Microsoft public program for developers that provides early access to unreleased features in Teams. This allows to explore and test upcoming features for potential inclusion in Microsoft Teams. They are provided for testing and exploration purposes only. They should not be used in production applications.

The Developer Preview can be enabled per Teams Client; thus, it doesn't affect the entire organization, only the instance (Teams Desktop or Teams Web application) where it is activated.

To activate the Developer Preview on a computer or web client, the uploading of apps must be activated as described at the beginning of the section. Click on the profile button (upper right corner of the Teams interface, the button with the picture of the user) to display the Teams menu, and then select About and click on Developer preview to turn it on or off.

The manifest used for customized components must have the property manifestVersion with the value devPreview. The functionality available changes very often and sometimes it is not documented by Microsoft.

Using the devPreview schema disallows the use of App Studio and the possibility to upload apps for testing. To upload an application, click the More apps icon on the app bar, then select the Upload a custom app link. This method only permits to upload a zipped version of the app package.

14.3.4. ngrok

To load custom Team apps, the app must be available from the internet; it cannot be used running from a local IIS. There are two possibilities to make the Teams app in development reachable from the internet: Hosting the app in a public server, such as Microsoft Azure, or creating a tunnel to the local process on the development machine using ngrok, an application (available for Windows, Linux and Mac) that creates public URLs for testing of software that runs in a local development computer.

Tunneling using ngrok is valid for testing running the app on the local machine, and creates a tunnel to it through a public web endpoint, but it is not suitable for production. When using the Teams App Studio to create the manifest, a message will appear indicating this. The message can be voided for testing, but not when the application will be deployed for production.

ngrok is a free tool that can be downloaded from https://ngrok.com/download. Unzip ngrok to a directory in the development computer. Run the Teams app under development from Visual Studio: The app will be available locally from a URL like http://localhost:3333. Open a PowerShell console, relocate the pointer to the folder where ngrok is unzipped, and run it using the syntax:

```
.\ngrok.exe http 3333 -host-header=localhost:3333
```

Ensure that you use the same port for the http parameter and the localhost parameter. The console will respond indicating the external URL generated from ngrok in the form of http(s)://[identifier].ngrok.io. For the free version of ngrok, a session can expand for max 8 hours (it is not necessary to register in the ngrok site to use the free version). From this moment, the application available locally under the URL http://localhost:3333 will be also available from the public internet URL http(s)://[identifier].ngrok.io. To stop the tunnel, use the command Ctrl-c.

ngrok provides a real-time web user interface as well, to gather all the HTTP traffic running over the tunnel. After starting ngrok, open the URL given under Web Interface in the PowerShell window (for example, http://127.0.0.1:4040) in a web browser to review all the traffic details.

To host Webs for testing or production, the Azure Web Apps Service (https://azure.microsoft.com/en-us/services/app-service/web/) provides ready to use hosting environments, where all the infrastructure is delivered by Microsoft. There are diverse price tiers, including one for free.

14.3.5. Cards

Cards are an open format that enables developers to exchange content for user interfaces in a commonly and consistently way. Cards are used in messages, bots, emails, and any kind of application that needs to show information for users. There are eight types of Cards available for Teams: Adaptive, Hero, List, Office 365 Connector, Receipt, Signin, Thumbnail, and Collections. No type can be used for any other type of application: Teams Connectors, for example, only accept Cards of the Office 365 Connector type.

Cards are described as JSON objects with a defined syntax. Microsoft provides extensive information about Adaptive Cards in its site https://docs.microsoft.com/en-us/adaptive-cards/.

The Teams App Studio tool (see section 3.1 in this chapter) contains a section to create the JSON for Hero, Thumbnail, and Adaptive Cards. It can inclusively generate the CSharp code to insert directly in the code for Teams apps.

For Message and Adaptive Cards, the Microsoft site https://messagecardplayground.azurewebsites.net offers several examples showing the JSON code and the Card result. This site is becoming replaced by the Microsoft site https://amdesigner.azurewebsites.net, that also has several examples and a Card generator but only for Adaptive Cards.

Working from CSharp, it is easier to use the NuGet AdaptiveCards than to parse JSON code manually (https://www.nuget.org/packages/AdaptiveCards/). This is a library that implements classes for building and serializing Adaptive Card objects from code (only for Adaptive Cards).

14.4. Teams Tabs

Tabs are Web pages embedded in Microsoft Teams. There are two types of Tabs available in Teams:

- Personal Tabs are scoped to a single user. They are pinned to the left navigation bar, under the ellipse (…) button.
- Channel/Group Tabs deliver content to channels and group chats. They are pinned to the top-horizontal bar (Tabs bar).

The Web pages to be used for Tabs must be hosted as HTTPS (secure socket layers) and able to be embedded in an iFrame by the Teams client.

14.4.1. Personal Tabs

Fundamentally, Tabs with a personal scope consist of Web pages that are framed within the Teams client, and that are accessible after installation from the ellipse menu at the left side of the Teams interface.

Any public Web page can be set as Personal Tab. For the next example, Visual Studio is used to create a .NET Framework ASP.NET Web Forms application, but an MVC application using .NET Framework or .NET Core would suffice as well.

Start Visual Studio and create a new solution of the type ASP.NET Web Application (.Net Framework). Select Empty as the type project, select Web Forms in the Core references menu, and deselect Configure for HTTPS.

Add a new Web Form called GenerateGuid.aspx to the solution. This will be the page with the functionality for the Tab. There are one button and two labels in the aspx page, one for the new GUID and other to show information from the context. The styles from the .css file are used for styling.

14.001	ID	File

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="GenerateGuid.aspx.cs"
    Inherits="KKJA.GenerateGuid" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="https://statics.teams.microsoft.com/sdk/v1.0/js/MicrosoftTeams.min.js"
        type="text/javascript"></script>
    <script src="GenerateAppScripts.js" type="text/javascript"></script>
    <link rel="stylesheet" href="GenerateThemes.css" type="text/css" />
</head>
<body class="theme-light">
    <form id="form1" runat="server">
        <div class="surface font-semibold font-title"><h2>Generate a new GUID</h2></div>
        <div>
            <p>
                <asp:Button ID="btnGenerateGuid" runat="server" Text="Generate"
                    OnClick="btnGenerateGuid_Click" />
            </p>
            <p class="surface">
                <asp:Label ID="lblNewGuid" runat="server" Text=""></asp:Label>
                <asp:Label ID="lblContextInfo" runat="server" Text=""></asp:Label>
            </p>
        </div>
    </form>
```

```
</body>
</html>
```

The .aspx page has a reference to the MicrosoftTeams.min.js file from the team's Content Distribution Network (CDN). This file belongs to the Team's JavaScript client SDK, a part of the Microsoft Teams developer platform, and it contains methods to facilitate the integration of custom services with Teams. There is also a reference to the custom stylesheet file GenerateThemes.css that contains all the styling classes for the Teams themes, and a reference to the GenerateAppScripts.js containing the JavaScript routine that initializes the Team's client SDK, checks the initial theme chosen by the user and maintains it applied, defines the event handler for the change of themes, and sets a theme when the change of theme event is detected. The context contains some information about Teams, the user and the session; the label lblContextInfo shows, for example, the value of the loginHint property present in the context.

14.002	ID		File	
Other Modules	MicrosoftTeams.min.js			

```
(function () {
    'use strict';

    microsoftTeams.initialize();

    microsoftTeams.getContext(function (context) {
        if (context && context.theme) {
            document.getElementById('lblContextInfo').innerText = context.loginHint;
            setTheme(context.theme);
        }
    });

    microsoftTeams.registerOnThemeChangeHandler(function (theme) {
        setTheme(theme);
    });

    function setTheme(theme) {
        if (theme) {
        // Possible values for theme: 'default', 'light', 'dark' and 'contrast'
        document.body.className = 'theme-' + (theme === 'default' ? 'light' : theme);
        }
    }
})();
```

The GenerateThemes.css file is too long (more than 1200 lines code) to be printed in the book. You can find it in the KKJA repo in the book's GitHub site.

The code-behind file for the .aspx page generates a GUID when the button is used and shows its value in the label.

14.003	ID		File
Routines			
NuGets			
Ref. DLLs			
Using			

```
protected void btnGenerateGuid_Click(object sender, EventArgs e)
{
    lblNewGuid.Text = Guid.NewGuid().ToString();
}
```

Two other aspx pages are necessary: One to show the privacy statement, and another for the terms of use. In this example, there is only some text in the pages, but any kind of functionality can be used.

14.004	ID		File

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Privacy.aspx.cs"
    Inherits="KKJA.Privacy" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="https://statics.teams.microsoft.com/sdk/v1.0/js/MicrosoftTeams.min.js"
        type="text/javascript"></script>
    <script src="GenerateAppScripts.js" type="text/javascript"></script>
    <link rel="stylesheet" href="GenerateThemes.css" type="text/css" />
</head>
<body class="theme-light">
    <form id="form1" runat="server">
        <div>
            This is the Privacy Statement page
        </div>
    </form>
</body>
</html>
```

14.005	ID	File

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Terms.aspx.cs"
Inherits="KKJA.Terms" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="https://statics.teams.microsoft.com/sdk/v1.0/js/MicrosoftTeams.min.js"
        type="text/javascript"></script>
    <script src="GenerateAppScripts.js" type="text/javascript"></script>
    <link rel="stylesheet" href="GenerateThemes.css" type="text/css" />
</head>
<body class="theme-light">
    <form id="form1" runat="server">
        <div>
            This is the Terms of Use page
        </div>
    </form>
</body>
</html>
```

Run the project from Visual Studio and take note of the port used by IIS Express. Start ngrok as indicated in section 4.3 of this chapter, using the port number from IIS Express.

Ensure that you are using the HTTP port of IIS when the project is running, and not the HTTPS because ngrok forwards the request to the HTTP port.

Open Teams (the desktop or web client) and open the App Studio. Open the Manifest editor tab and click on the Create a new app button. In the App details section, define the Short name and Long name of the application (any combination of strings) and click on the Generate button under the Identification section. Then define a Package Name and Version, Description, Long description, Name and Website of the developer. In the App URLs box copy the ngrok URL extended with the file names of the privacy and terms pages.

Click on the Tabs button under Capabilities, and then on the Add button for Add a personal tab. On the new window, define a Name for the tab, a unique string for the Entity ID (it can be any string, but it must be unique), and add the ngrok URL for the content page of the application in the Content URL and Website URL boxes.

Finally, click on the Test and distribute button under the Finish section. If there are errors in the

information, the validation will show them. Use the Install button, and Teams will show the Personal Tab application. The manifest can be downloaded, and it will be like the next one.

```
{
    "$schema": "https://developer.microsoft.com/en-us/json-schemas/teams
/v1.5/MicrosoftTeams.schema.json",
    "manifestVersion": "1.5",
    "version": "1.0.0",
    "id": "48cb3b67-6afb-49e2-be45-d3bdc34aef10",
    "packageName": "bookPersonalTab",
    "developer": {
        "name": "gavd",
        "websiteUrl": "https://43b609a6.ngrok.io/generateguid.aspx",
        "privacyUrl": "https://43b609a6.ngrok.io/privacy.aspx",
        "termsOfUseUrl": "https://43b609a6.ngrok.io/terms.aspx"
    },
    "icons": {
        "color": "color.png",
        "outline": "outline.png"
    },
    "name": {
        "short": "PersonalTab",
        "full": "Personal Tab for the book"
    },
    "description": {
        "short": "Personal Tab for the book",
        "full": "This is the Personal Tab for the book"
    },
    "accentColor": "#FFFFFF",
    "staticTabs": [
    {
        "entityId": "guidGenerator01",
        "name": "GuidGenerator",
        "contentUrl": "https://43b609a6.ngrok.io/generateguid.aspx",
        "websiteUrl": "https://43b609a6.ngrok.io/generateguid.aspx",
        "scopes": [
            "personal"
        ]
    }
    ],
    "permissions": [
        "identity",
        "messageTeamMembers"
    ],
        "validDomains": [
        "43b609a6.ngrok.io"
    ]
}
```

Office 365:

The best recipes for developers

This is a book dedicated to coders: It explains how to work programmatically with Microsoft Office 365, the collaboration and information sharing platform of Microsoft.

Office 365 offers a few servers (Exchange, SharePoint), editing and authoring tools (Outlook, Word, Excel, PowerPoint), and a myriad of other applications to help businesses in creating, managing and organizing information.

This is a book made by and targeted to developers. We assume the readers do know how the Office applications work: You will not find functional descriptions or instructions for users, but countless code routines and programming methods. The book is also for developers that know the programming tools and technologies used by Microsoft and Office 365: Visual Studio, Visual Studio Code, CSharp, PowerShell, JavaScript, etc.

About the Author

Gustavo Velez is a mechanical and electronics engineer working as a software developer and senior solutions architect for more than thirty years. Specialized in integration of Microsoft software, he started working with SharePoint before the server got its current name: in 1998, Gustavo finalized his first enterprise collaboration project using Site Server, the precursor of SharePoint.

Gustavo is Microsoft's Most Valuable Professional (MVP) since 2008. In his many years of experience developing and working with Windows and Office applications, Gustavo has given seminars/training in SharePoint and has also done consultancy work for several of the biggest SharePoint implementations in Europe, Africa, and South America. His articles can be found in many of the leading trade magazines in English, Dutch, German, and Spanish. Gustavo is author and co-author of ten books about SharePoint, and founder and editor of CompartiMOSS (http://www.compartimoss.com), a highly read magazine about Microsoft technologies for the Spanish-speaking community.

About Güitaca Publishers

Güitaca Publishers is an independent company dedicated to bringing out technical books in electronic format, especially about Information Technology, computer developing, and systems engineering. We aim to present the most recent publications to our readers using novel processes as, for example, our "Books by subscription" concept.

güitaca publishers