

# Office 365:

*The best recipes  
for developers*

Gustavo Velez



**BOOK ONLY  
(no Subscription)**

First Edition

# Office 365 - The best recipes for developers

Copyright © 2020 Gütaca Publishers (<https://guitaca.com>)

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author nor Gütaca Publishers, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

This book is sold with the understanding that neither the publisher nor the author is engaged in providing legal, accounting, or other professional service, and that they shall not be liable for damages arisen from the use of the published code.

First published: April 2020

ISBN 978-958-48-8932-4



*Amsterdam - Bogotá - Bilbao*

# About Gütaca Publishers

Gütaca 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. Gütaca, coming from the Muisca mythology, is our symbol and guardian. The Muisca 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.

Gütaca 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 Gütaca 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 Gütaca 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***

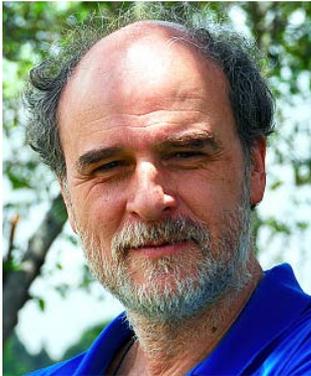
Alejandro and Jordi are the creators of our covers, logos and all the artwork of Gütaca. Alejandro has developed his career as copywriter, account executive and project coordinator in his [bilbaosolutions.com](http://bilbaosolutions.com) own company. Jordi is trained in artistic disciplines and computer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

If you wish to make contact with the Gütaca Publishers team, please send us an email to [info@guitaca.com](mailto: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 [Comparti MOSS](#) 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 Gütaca 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 Gütaca 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.websuperqoo.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.

	<p>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:</p> <p><a href="https://developer.microsoft.com/en-us/office/dev-program">https://developer.microsoft.com/en-us/office/dev-program</a>.</p> <p>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.</p>
---	---

## *Books by subscription*

This is one in the series of books published by Gütaca 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 Gütaca (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](mailto: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 Gütaca), 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).

	<p><b><i>Data privacy statement:</i></b></p> <p>We only ask for an email address for registration of your book's copy and make the downloads available for you. <b>We DON'T use it for any commercial purpose, and we will never monetize it in any way.</b> You will only receive a few emails from us:</p> <ul style="list-style-type: none"><li>- When you buy the book (or subscription) from our site, you receive one email with the acknowledgement of receipt and your License Key code.</li><li>- 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.</li><li>- 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).</li><li>- If your subscription is coming to an end, we will send you one email with instructions on how to renew it.</li></ul>
---	---

## *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.



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

---

<b>1</b>	<b>Exchange Online</b>	<b>1</b>
1.1.	<i>Introduction to developing for Exchange</i>	2
1.2.	<i>Login in Exchange</i>	2
1.2.1.	<i>Login from a managed language (CSharp)</i>	2
1.2.2.	<i>Login from PowerShell</i>	3
1.2.3.	<i>Login with user credentials for EWS (Basic Authentication) and CSharp</i>	4
1.2.4.	<i>Login with oAuth for EWS and CSharp</i>	6
1.2.5.	<i>Login for EWS with PowerShell and Basic Authentication</i>	8
1.2.6.	<i>Login for EWS with PowerShell and oAuth</i>	9
1.2.7.	<i>Login using the Exchange Online PowerShell</i>	10
1.3.	<i>Programming Exchange with EWS and CSharp</i>	11
1.3.1.	<i>Folders and folder structure using EWS</i>	12
1.3.2.	<i>Working with emails</i>	18
1.3.3.	<i>Using contacts in Exchange</i>	36
1.3.4.	<i>Developing for calendars in Exchange with EWS</i>	43
1.4.	<i>Programming Exchange with EWS and PowerShell</i>	57
1.4.1.	<i>Working with Exchange Folders using EWS and PowerShell</i>	57
1.4.2.	<i>Using emails with EWS and PowerShell</i>	60
1.4.3.	<i>Programming the Exchange contacts with PowerShell and EWS</i>	63
1.4.4.	<i>Approaching the Calendar with EWS and PowerShell</i>	67
1.5.	<i>Tools to work with EWS</i>	70
1.6.	<i>Using Exchange Online PowerShell</i>	71

---

<b>2</b>	<b>Microsoft Word Online</b>	<b>75</b>
2.1.	<i>Introduction to developing for the Office 365 clients</i>	76
2.2.	<i>Introduction to OpenXML for Word</i>	77
2.3.	<i>Working with Word documents and OpenXML (CSharp)</i>	79
2.3.1.	<i>Creation of Word documents with OpenXML</i>	79
2.3.2.	<i>Read and add text to a Word document</i>	80
2.3.3.	<i>The Word document properties</i>	82
2.3.4.	<i>Creating and adding styles in a Word document</i>	84

2.3.5.	<i>Word document headers and footers</i>	87
2.3.6.	<i>Working with comments in Word documents</i>	92
2.3.7.	<i>Tables and content in Word documents</i>	95
2.3.8.	<i>Word documents with images</i>	98
2.3.9.	<i>Other operations with OpenXML and Word documents</i>	102
2.4.	<i>Introduction to Word Visual Studio Tools for Office (VSTO)</i>	106
2.5.	<i>Developing VSTO applications for Word Office 365</i>	106
2.5.1.	<i>Basic Word VSTO Add-in</i>	106
2.5.2.	<i>Modifying the Ribbon with a Word VSTO Add-in</i>	107
2.5.3.	<i>Basic VSTO document-level customization</i>	110
2.6.	<i>Introduction to Word Web Add-ins</i>	110
2.7.	<i>Developing Word Web Add-ins for Office 365 with Visual Studio</i>	111
2.7.1.	<i>Basic Web Add-in for Word Office 365</i>	111
2.7.2.	<i>Word Web Add-in calling a REST service</i>	113

---

<b>3</b>	<b>Microsoft Excel Online</b>	<b>116</b>
3.1.	<i>Introduction</i>	117
3.1.1.	<i>Different forms of Excel Add-Ins</i>	117
3.1.2.	<i>The Developer Tab in Office 365</i>	117
3.2.	<i>Introduction to OpenXML</i>	118
3.3.	<i>Working with Excel spreadsheets and OpenXML (CSharp)</i>	119
3.3.1.	<i>Creation of Excel spreadsheets with OpenXML</i>	120
3.3.2.	<i>Add cells and values to spreadsheets with OpenXML</i>	120
3.3.3.	<i>Read the cell values in a spreadsheet</i>	122
3.3.4.	<i>Update a cell value in a spreadsheet</i>	127
3.3.5.	<i>Find all sheets in a spreadsheet</i>	128
3.3.6.	<i>Find hidden columns and rows in a spreadsheet</i>	130
3.3.7.	<i>Adding charts to a spreadsheet</i>	131
3.4.	<i>Introduction to Excel Visual Studio Tools for Office (VSTO)</i>	136
3.5.	<i>Developing VSTO applications for Excel Office 365</i>	137
3.5.1.	<i>Basic Excel VSTO Add-in</i>	137
3.5.2.	<i>Adding and using panels with a VSTO Add-in for Excel</i>	138
3.5.3.	<i>Basic VSTO Excel document-level customization</i>	141
3.6.	<i>Introduction to Excel Web Add-ins</i>	142
3.7.	<i>Developing Excel Web Add-ins for Office 365 with Visual Studio</i>	142
3.7.1.	<i>Basic Web Add-ins for Excel Office 365</i>	142
3.7.2.	<i>Excel Web Add-in calling a REST service</i>	146

3.8. Using external libraries to work with Excel	148
3.8.1. Create a new Excel spreadsheet and set cells using EPPlus	148
3.8.2. Read the value of one cell	149
3.8.3. Update the cell values with EPPlus	150
3.8.4. Creation of charts	150
3.8.5. Adding graphics and styles to spreadsheets	153
3.8.6. Working with formulas in spreadsheets using EPPlus	154

---

<b>4 Microsoft PowerPoint Online</b>	<b>156</b>
4.1. Introduction	157
4.2. Introduction to OpenXML	157
4.3. Working with PowerPoint presentation and OpenXML (CSharp)	159
4.3.1. Creation of PowerPoint presentations with OpenXML	159
4.3.2. Find text in one slide	168
4.3.3. Change the theme of a presentation	172
4.3.4. Working with slides in a PowerPoint presentation	173
4.3.5. Working with comments in presentations	179
4.3.6. Working with notes in slides	183
4.3.7. Add pictures and shapes to a slide in the presentation	185
4.4. Introduction to Visual Studio Tools for Office (VSTO) and PowerPoint	188
4.5. Developing customizations for PowerPoint with VSTO	188
4.5.1. Basic PowerPoint VSTO Add-in	189
4.6. Introduction to PowerPoint Web Add-ins	190
4.7. Developing Web Add-ins for PowerPoint presentations	190
4.7.1. Basic Web Add-ins for PowerPoint	190
4.7.2. PowerPoint presentations getting data from a REST service	193

---

<b>5 Microsoft Outlook Online</b>	<b>196</b>
5.2. Using Outlook Visual Studio Tools for Office (VSTO)	197
5.2.1. Introduction of VSTO for Outlook	197
5.2.2. Limitations of VSTO in Outlook	197
5.2.3. Basic VSTO Add-in applications for Outlook 365	198
5.2.4. Modifying the Outlook ribbon	200
5.2.5. Working with emails and VSTO	201
5.2.6. Outlook Contacts through VSTO	203

5.2.7. <i>Programming the Outlook Calendar with VSTO</i>	205
5.2.8. <i>Folders in Outlook and VSTO</i>	207
<b>5.3. Outlook Web Add-ins</b>	<b>209</b>
5.3.1. <i>Introduction to Outlook Web Add-ins</i>	209
5.3.2. <i>Basic Outlook Web Add-in</i>	210
5.3.3. <i>Getting information from a REST service inside an Outlook Add-in</i>	212

---

<b>6 SharePoint Introduction and Login Routines</b>	<b>214</b>
6.1. <i>Login (CSharp) using the SharePoint Client Object Model (CSOM)</i>	215
6.2. <i>Login (CSharp) using PnP Core</i>	217
6.3. <i>Login (CSharp) to use REST</i>	218
6.4. <i>Login (PowerShell) using CSOM</i>	222
6.5. <i>Login (PowerShell) using the SharePoint Online SPO cmdlets</i>	224
6.6. <i>Login (PowerShell) using PnP for SharePoint Online</i>	225
6.7. <i>Login (PowerShell) to use REST</i>	226

---

<b>7 SharePoint Online Tenant</b>	<b>232</b>
7.1. <i>Introduction</i>	233
7.2. <i>Working with the tenant and the Client Side Object Model (CSharp)</i>	233
7.2.1. <i>Retrieve the tenant properties configuration - CSOM, CSharp</i>	233
7.2.2. <i>Update the tenant properties configuration - CSOM, CSharp</i>	234
7.3. <i>Approaching the tenant using REST (CSharp)</i>	235
7.3.1. <i>Find the App Catalog URL - REST, CSharp</i>	235
7.3.2. <i>Find tenant properties - REST, CSharp</i>	235
7.4. <i>Working in the tenant using PowerShell CSOM (PowerShell)</i>	236
7.4.1. <i>Tenant properties configuration - CSOM, PowerShell</i>	236
7.4.2. <i>Update the tenant properties configuration - CSOM, PowerShell</i>	237
7.5. <i>Approaching the tenant using REST (PowerShell)</i>	237
7.5.1. <i>Find the App Catalog URL - REST, PowerShell</i>	237
7.5.2. <i>Find tenant properties - REST, PowerShell</i>	238
7.6. <i>Using SPO cmdlets for the tenant (PowerShell)</i>	238
7.6.1. <i>Retrieve and modify tenant properties - SPO, PowerShell</i>	238
7.6.2. <i>Get the tenant error logs - SPO, PowerShell</i>	239
7.6.3. <i>Working with the CDN - SPO, PowerShell</i>	240

<b>8</b>	<b>SharePoint Online Site Collections and Webs</b>	<b>245</b>
8.1.	<i>Introduction</i>	246
8.2.	<i>Operations for modern Site Collections with the Client Side Object Model (CSharp)</i>	246
8.2.1.	<i>Creation of modern Site Collections - CSOM, CSharp</i>	246
8.2.2.	<i>Enumeration of Site Collections in the Tenant - CSOM, CSharp</i>	248
8.2.3.	<i>Delete Site Collections from the Tenant - CSOM, CSharp</i>	249
8.2.4.	<i>Add users with rights to one Site Collection - CSOM, CSharp</i>	250
8.2.5.	<i>Working with modern Hub Sites - CSOM, CSharp</i>	251
8.3.	<i>Operations for Webs in a Site Collection with the Client Side Object Model (CSharp)</i>	254
8.3.1.	<i>Create Web Sites in a Site Collection - CSOM, CSharp</i>	254
8.3.2.	<i>Find the Webs of a Site Collection - CSOM, CSharp</i>	255
8.3.3.	<i>Update one Web in a Site Collection - CSOM, CSharp</i>	256
8.3.4.	<i>Delete one Web from a Site Collection - CSOM, CSharp</i>	256
8.3.5.	<i>Break and Reset the security inheritance of a Web Site - CSOM, CSharp</i>	257
8.3.6.	<i>Add, update and delete users to the security configuration of a Web Site - CSOM, CSharp</i>	258
8.4.	<i>CRUD operations for Site Collections and Webs using PnP Core (CSharp)</i>	260
8.4.1.	<i>Create Site Collections and Web sites - PnPCore, CSharp</i>	260
8.4.2.	<i>Enumerate all Webs in a Site Collection - PnPCore, CSharp</i>	262
8.4.3.	<i>Operations with PnPCore for Site Collections and Webs - PnPCore, CSharp</i>	262
8.5.	<i>CRUD operations using REST for Site Collections and Webs (CSharp)</i>	263
8.5.1.	<i>Creating Site Collections and Webs inside Site Collections - REST, CSharp</i>	263
8.5.2.	<i>Enumerate Site Collections and Webs - REST, CSharp</i>	266
8.5.3.	<i>Update the properties of a Web - REST, CSharp</i>	267
8.5.4.	<i>Delete Webs from Site Collections - REST, CSharp</i>	268
8.5.5.	<i>Permissions in a Web - REST, CSharp</i>	268
8.5.6.	<i>Break and reset the security inheritance of Webs - REST, CSharp</i>	270
8.5.7.	<i>Add, update and delete users in Webs - REST, CSharp</i>	271
8.6.	<i>Working with Site Collections using PowerShell CSOM(PowerShell)</i>	274
8.6.1.	<i>Creation of modern Site Collections - CSOM, PowerShell</i>	275
8.6.2.	<i>Enumeration of Site Collections in the Tenant - CSOM, PowerShell</i>	277
8.6.3.	<i>Delete Site Collections from the Tenant - CSOM, PowerShell</i>	277
8.6.4.	<i>Add users with rights to one Site Collection - CSOM, PowerShell</i>	279

8.6.5.	<i>Working with modern Hub Sites - CSOM, PowerShell</i>	279
8.7.	<b><i>PowerShell CSOM used to work with Webs (PowerShell)</i></b>	<b>282</b>
8.7.1.	<i>Create Web Sites in a Site Collection - CSOM, PowerShell</i>	282
8.7.2.	<i>Find the Webs of a Site Collection - CSOM, PowerShell</i>	283
8.7.3.	<i>Update one Web in a Site Collection - CSOM, PowerShell</i>	284
8.7.4.	<i>Delete one Web from a Site Collection - CSOM, PowerShell</i>	285
8.7.5.	<i>Break and Reset the security inheritance of a Web Site - CSOM, PowerShell</i>	285
8.7.6.	<i>Add, update and delete users to the security configuration of a Web Site - CSOM, PowerShell</i>	286
8.8.	<b><i>Operations for Site Collections using PowerShell PnP (PowerShell)</i></b>	<b>288</b>
8.8.1.	<i>Create Site Collections - PnP, PowerShell</i>	288
8.8.2.	<i>Retrieve Site Collections - PnP, PowerShell</i>	289
8.8.3.	<i>Update Site Collections - PnP, PowerShell</i>	290
8.8.4.	<i>Delete Site Collections - PnP, PowerShell</i>	291
8.8.5.	<i>Working with Hub Site Collections - PnP, PowerShell</i>	292
8.8.6.	<i>Rights and Permissions for Site Collections - PnP, PowerShell</i>	293
8.9.	<b><i>Operations for Webs using PowerShell PnP (PowerShell)</i></b>	<b>294</b>
8.9.1.	<i>Creation of Webs in Site Collections - PnP, PowerShell</i>	294
8.9.2.	<i>Enumerate Webs in a Site Collection - PnP, PowerShell</i>	295
8.9.3.	<i>Update properties in Webs - PnP, PowerShell</i>	295
8.9.4.	<i>Deleting Web from Site Collections - PnP, PowerShell</i>	296
8.9.5.	<i>Security permissions for a Web - PnP, PowerShell</i>	296
8.10.	<b><i>Operations for Site Collections using SharePoint Online (SPO) cmdlets (PowerShell)</i></b>	<b>296</b>
8.10.1.	<i>Create, test and repair Site Collections - SPO, PowerShell</i>	297
8.10.2.	<i>Find Site Collections - SPO, PowerShell</i>	298
8.10.3.	<i>Update Site Collections - SPO, PowerShell</i>	299
8.10.4.	<i>Delete Site Collections - SPO, PowerShell</i>	299
8.10.5.	<i>Working with Hub Site Collections - SPO, PowerShell</i>	300
8.10.6.	<i>Security-related cmdlets for users - SPO, PowerShell</i>	303
8.10.7.	<i>Security-related cmdlets for groups - SPO, PowerShell</i>	305
8.11.	<b><i>CRUD operations for Site Collections and Webs using PowerShell REST (PowerShell)</i></b>	<b>307</b>
8.11.1.	<i>Creating Site Collections and Webs inside Site Collections - REST, PowerShell</i>	307
8.11.2.	<i>Enumerate the Site Collections and Webs - REST, PowerShell</i>	309
8.11.3.	<i>Update methods for a Web - REST, PowerShell</i>	310
8.11.4.	<i>Delete Webs from Site Collections - REST, PowerShell</i>	310
8.11.5.	<i>Permissions for users in a Web - REST, PowerShell</i>	311
8.11.6.	<i>Break and reset the security inheritance of Webs - REST, PowerShell</i>	312
8.11.7.	<i>Add, update and delete users to the security of Webs - REST, PowerShell</i>	313

<b>9</b>	<b>SharePoint Online Lists and Libraries</b>	<b>317</b>
9.1.	<i>Introduction</i>	318
9.2.	<i>CRUD operations for Lists with the Client Side Object Model (CSharp)</i>	318
9.2.1.	<i>List creation - CSOM, CSharp</i>	318
9.2.2.	<i>List find and read properties - CSOM, CSharp</i>	319
9.2.3.	<i>List Update - CSOM, CSharp</i>	320
9.2.4.	<i>List Delete - CSOM, CSharp</i>	320
9.2.5.	<i>Add one Field to a List - CSOM, CSharp</i>	321
9.2.6.	<i>Read the Fields in a List - CSOM, CSharp</i>	321
9.2.7.	<i>Update one List Field - CSOM, CSharp</i>	322
9.2.8.	<i>Eliminate one Field from a List - CSOM, CSharp</i>	323
9.2.9.	<i>Break and Reset the List's Security Inheritance - CSOM, CSharp</i>	323
9.2.10.	<i>Add one user with permissions to the List's Security - CSOM, CSharp</i>	325
9.2.11.	<i>Update the user permissions in the List's Security - CSOM, CSharp</i>	325
9.2.12.	<i>Delete one user from the Security for the List - CSOM, CSharp</i>	326
9.3.	<i>CRUD operations for Lists with PnPCore (CSharp)</i>	326
9.3.1.	<i>List creation - PnPCore, CSharp</i>	327
9.3.2.	<i>List find and read properties - PnPCore, CSharp</i>	327
9.3.3.	<i>List Exists - PnPCore, CSharp</i>	328
9.3.4.	<i>Add one Field to a List - PnPCore, CSharp</i>	328
9.3.5.	<i>Read the Fields in a List - PnPCore, CSharp</i>	329
9.3.6.	<i>Add permissions to the List - PnPCore, CSharp</i>	330
9.3.7.	<i>Get one Content Type used in the List - PnPCore, CSharp</i>	331
9.3.8.	<i>Add one Content Type to a List - PnPCore, CSharp</i>	331
9.3.9.	<i>Eliminate one Content Type from a List - PnPCore, CSharp</i>	332
9.3.10.	<i>Find one View for a List - PnPCore, CSharp</i>	332
9.3.11.	<i>Create Views for Lists - PnPCore, CSharp</i>	333
9.4.	<i>CRUD operations for Lists with REST (CSharp)</i>	333
9.4.1.	<i>List creation - REST, CSharp</i>	333
9.4.2.	<i>List find and read properties - REST, CSharp</i>	334
9.4.3.	<i>List Update - REST, CSharp</i>	335
9.4.4.	<i>List Delete - REST, CSharp</i>	335
9.4.5.	<i>Add one Field to a List - REST, CSharp</i>	336
9.4.6.	<i>Read the Fields in a List - REST, CSharp</i>	337
9.4.7.	<i>Update one List Field - REST, CSharp</i>	338
9.4.8.	<i>Eliminate one Field from a List - REST, CSharp</i>	339
9.4.9.	<i>Break and Reset the List's Security Inheritance - REST, CSharp</i>	339

9.4.10.	<i>Add one user with permissions to the List's Security - REST, CSharp</i>	340
9.4.11.	<i>Update the user permissions in the List's Security - REST, CSharp</i>	342
9.4.12.	<i>Delete one user from the Security for the List - REST, CSharp</i>	343
<b>9.5.</b>	<b><i>CRUD operations for Lists with PowerShell CSOM (PowerShell)</i></b>	<b>344</b>
9.5.1.	<i>List creation - CSOM, PowerShell</i>	344
9.5.2.	<i>Find a List and read its properties - CSOM, PowerShell</i>	345
9.5.3.	<i>List Update - CSOM, PowerShell</i>	346
9.5.4.	<i>List Delete - CSOM, PowerShell</i>	346
9.5.5.	<i>Add one Field to a List - CSOM, PowerShell</i>	346
9.5.6.	<i>Retrieve the Fields in a List - CSOM, PowerShell</i>	347
9.5.7.	<i>Update one List Field - CSOM, PowerShell</i>	348
9.5.8.	<i>Eliminate one Field from a List - CSOM, PowerShell</i>	349
9.5.9.	<i>Break and Reset the List's Security Inheritance - CSOM, PowerShell</i>	349
9.5.10.	<i>Add one user with permissions to the List's Security - CSOM, PowerShell</i>	351
9.5.11.	<i>Update the user permissions in the List's Security - CSOM, PowerShell</i>	352
9.5.12.	<i>Delete one user from the Security for the List - CSOM, PowerShell</i>	353
<b>9.6.</b>	<b><i>CRUD operations for Lists with PnP PowerShell (PowerShell)</i></b>	<b>353</b>
9.6.1.	<i>List creation - PnP, PowerShell</i>	353
9.6.2.	<i>List find and read properties - PnP, PowerShell</i>	354
9.6.3.	<i>List Update - PnP, PowerShell</i>	355
9.6.4.	<i>List Delete - PnP, PowerShell</i>	355
9.6.5.	<i>Add one Field to a List - PnP, PowerShell</i>	355
9.6.6.	<i>Read the Fields in a List - PnP, PowerShell</i>	356
9.6.7.	<i>Update one List Field - PnP, PowerShell</i>	357
9.6.8.	<i>Eliminate one Field from a List - PnP, PowerShell</i>	357
<b>9.7.</b>	<b><i>CRUD operations for Lists with REST and PowerShell (PowerShell)</i></b>	<b>358</b>
9.7.1.	<i>List creation - REST, PowerShell</i>	358
9.7.2.	<i>List find and read properties - REST, PowerShell</i>	358
9.7.3.	<i>List Update - REST, PowerShell</i>	359
9.7.4.	<i>List Delete - REST, PowerShell</i>	360
9.7.5.	<i>Add one Field to a List - REST, PowerShell</i>	360
9.7.6.	<i>Read the Fields in a List - REST, PowerShell</i>	361
9.7.7.	<i>Update one List Field - REST, PowerShell</i>	362
9.7.8.	<i>Eliminate one Field from a List - REST, PowerShell</i>	362
9.7.9.	<i>Break and Reset the List's Security Inheritance - REST, PowerShell</i>	363
9.7.10.	<i>Add one user with permissions to the List's Security - REST, PowerShell</i>	364
9.7.11.	<i>Update the user permissions in the List's Security - REST, PowerShell</i>	365
9.7.12.	<i>Delete one user from the Security for the List - REST, CSharp</i>	366

<b>10</b>	<b>SharePoint Online Items and Documents</b>	<b>367</b>
10.1.	<i>Introduction</i>	368
10.2.	<i>Operations for Items and Documents with the Client Side Object Model (CSharp)</i>	368
10.2.1.	<i>Items creation - CSOM, CSharp</i>	368
10.2.2.	<i>Documents upload - CSOM, CSharp</i>	369
10.2.3.	<i>Documents download - CSOM, CSharp</i>	371
10.2.4.	<i>Find Items and read their properties - CSOM, CSharp</i>	373
10.2.5.	<i>Find Files and read their properties - CSOM, CSharp</i>	374
10.2.6.	<i>Update List Items - CSOM, CSharp</i>	376
10.2.7.	<i>Update Document properties - CSOM, CSharp</i>	376
10.2.8.	<i>Delete List Items - CSOM, CSharp</i>	377
10.2.9.	<i>Delete Documents - CSOM, CSharp</i>	378
10.2.10.	<i>Break and reset the Item's and File's security inheritance - CSOM, CSharp</i>	379
10.2.11.	<i>Add one user with permissions to the Items and Documents - CSOM, CSharp</i>	380
10.2.12.	<i>Update the user permissions for Items and Documents - CSOM, CSharp</i>	381
10.2.13.	<i>Delete one user from the Item's and Document's Security - CSOM, CSharp</i>	382
10.3.	<i>Operations for Items and Documents with PnPCore (CSharp)</i>	382
10.3.1.	<i>Property Bag key creation - PnPCore, CSharp</i>	382
10.3.2.	<i>Reading a Property Bag key - PnPCore, CSharp</i>	383
10.3.3.	<i>Property Bag entry exists - PnPCore, CSharp</i>	383
10.3.4.	<i>Indexing and reading indexed Property Bag entries - PnPCore, CSharp</i>	384
10.3.5.	<i>Delete one Property Bag entry - PnPCore, CSharp</i>	384
10.4.	<i>Operations for Items and Documents with REST (CSharp)</i>	385
10.4.1.	<i>Items creation - REST, CSharp</i>	385
10.4.2.	<i>Documents upload - REST, CSharp</i>	386
10.4.3.	<i>Documents download - REST, CSharp</i>	386
10.4.4.	<i>Find Items and read their properties - REST, CSharp</i>	387
10.4.5.	<i>Find Documents in a Library and read their properties - REST, CSharp</i>	388
10.4.6.	<i>Update List Items and Files in a Library - REST, CSharp</i>	389
10.4.7.	<i>Delete List Items - REST, CSharp</i>	391
10.4.8.	<i>Break and reset the Item's and Document's security inheritance - REST, CSharp</i>	392
10.4.9.	<i>Add one user with permissions to the Item's and Document's Security - REST, CSharp</i>	393
10.4.10.	<i>Update the user permissions for the Item's and Document's Security - REST, CSharp</i>	394
10.4.11.	<i>Delete one user from the Item's and Document's Security - REST, CSharp</i>	396

10.5.	<i>Operations for Items and Documents with PowerShell CSOM (PowerShell)</i>	397
10.5.1.	<i>List Item creation - CSOM, PowerShell</i>	397
10.5.2.	<i>Documents upload - CSOM, PowerShell</i>	398
10.5.3.	<i>Documents download - CSOM, PowerShell</i>	399
10.5.4.	<i>Find Items and Files, and read their properties - CSOM, PowerShell</i>	400
10.5.5.	<i>Update List Items - CSOM, PowerShell</i>	402
10.5.6.	<i>Delete List Items - CSOM, PowerShell</i>	403
10.5.7.	<i>Break and reset the Item's and Document's security inheritance - CSOM, PowerShell</i>	405
10.5.8.	<i>Add one user with permissions to the Item's and Document's Security - CSOM, PowerShell</i>	406
10.5.9.	<i>Update the user permissions for the Item's and File's Security - CSOM, PowerShell</i>	407
10.5.10.	<i>Delete one user from the Item's and Document's Security - CSOM, PowerShell</i>	408
10.6.	<i>Operations for Items and Files with PnP PowerShell (PowerShell)</i>	408
10.6.1.	<i>Items creation - PnP, PowerShell</i>	408
10.6.2.	<i>Documents upload - PnP, PowerShell</i>	409
10.6.3.	<i>Documents download - PnP, PowerShell</i>	409
10.6.4.	<i>Find and enumerate Items - PnP, PowerShell</i>	410
10.6.5.	<i>Find, copy and move Files in Libraries - PnP, PowerShell</i>	410
10.6.6.	<i>Update List Items and Documents - PnP, PowerShell</i>	412
10.6.7.	<i>Delete List Items and Library Documents - PnP, PowerShell</i>	414
10.6.8.	<i>Security for Items and Documents - PnP, PowerShell</i>	415
10.7.	<i>Operations for Items and Documents with REST and PowerShell (PowerShell)</i>	416
10.7.1.	<i>Items creation - REST, PowerShell</i>	416
10.7.2.	<i>Documents upload - REST, PowerShell</i>	416
10.7.3.	<i>Documents download - REST, PowerShell</i>	417
10.7.4.	<i>Find Items and Files, and read their properties - REST, PowerShell</i>	417
10.7.5.	<i>Update List Items - REST, PowerShell</i>	419
10.7.6.	<i>Delete List Items and Library files - REST, PowerShell</i>	420
10.7.7.	<i>Break and reset the Item's and Document's security inheritance - REST, PowerShell</i>	421
10.7.8.	<i>Add one user with permissions to the Item's and Document's Security - REST, PowerShell</i>	422
10.7.9.	<i>Update the user permissions for the Item's and Document's Security - REST, PowerShell</i>	423
10.7.10.	<i>Delete one user from the Item's and Document's Security - REST, PowerShell</i>	424

<b>11</b>	<b>SharePoint Online - Other Components</b>	<b>426</b>
11.2.	<i>The Term Store</i>	427
11.2.1.	<i>Using the SharePoint Client Side Object Model programmatically to work with the Term Store (CSharp)</i>	427
11.2.2.	<i>Using PnPCore with the Term Store (CSharp)</i>	433
11.2.3.	<i>Using PowerShell and the CSOM for the Term Store (PowerShell)</i>	437
11.2.4.	<i>Using PowerShell PnP with the Term Store (PowerShell)</i>	443
11.3.	<i>Search</i>	447
11.3.1.	<i>Search and the SharePoint Client Side Object Model (CSharp)</i>	447
11.3.2.	<i>Using REST to access the SharePoint Search Engine (CSharp)</i>	448
11.3.3.	<i>Calling the Search Engine with PowerShell and the CSOM (PowerShell)</i>	449
11.3.4.	<i>PowerShell PnP to access the SharePoint Search Engine (PowerShell)</i>	449
11.3.5.	<i>PowerShell and REST to call the Search Engine (PowerShell)</i>	450
11.4.	<i>User Profile</i>	451
11.4.1.	<i>Approaching the User Profile with CSOM (CSharp)</i>	452
11.4.2.	<i>REST to access the User Profile (CSharp)</i>	454
11.4.3.	<i>Using CSOM PowerShell to reach the User Profile (PowerShell)</i>	456
11.4.4.	<i>Using PnP PowerShell to reach the User Profile (PowerShell)</i>	459
11.4.5.	<i>PowerShell and REST to access the User Profile (PowerShell)</i>	459
<b>12</b>	<b>SharePoint Online - SPFx</b>	<b>462</b>
12.1.	<i>Introduction</i>	463
12.2.	<i>SPFx WebParts</i>	464
12.2.1.	<i>Basic SPFx WebPart</i>	465
12.2.2.	<i>CRUD with SPFx WebParts (JavaScript and REST)</i>	469
12.2.3.	<i>Adding external libraries to an SPFx solution</i>	477
12.2.4.	<i>CRUD with SPFx WebParts (JavaScript and PnPjs)</i>	483
12.2.5.	<i>Other examples of PnPjs routines for SPFx WebParts</i>	490
12.3.	<i>SPFx Extensions</i>	497
12.3.1.	<i>Basic Application Customizer SPFx Extension</i>	497
12.3.2.	<i>Basic Field Customizer SPFx Extension</i>	499
12.3.3.	<i>Basic ListView Command Set SPFx Extension</i>	500
<b>13</b>	<b>SharePoint Online - Add-ins</b>	<b>505</b>

13.1.	<i>Introduction</i>	506
13.1.1.	<i>Developing SharePoint Add-ins</i>	506
13.2.	<i>SharePoint Hosted Add-ins</i>	507
13.2.1.	<i>Basic (immersive) SharePoint Hosted Add-in</i>	508
13.2.2.	<i>AppPart SharePoint Hosted Add-ins</i>	510
13.2.3.	<i>Custom Action SharePoint Hosted Add-ins</i>	511
13.2.4.	<i>Deploying SharePoint Hosted Add-ins to SharePoint Online</i>	513
13.3.	<i>Provider Hosted Add-ins</i>	513
13.3.1.	<i>Basic (immersive) Provider Hosted SharePoint Add-in</i>	513
13.3.2.	<i>Adding the chrome to a Provider Hosted Add-in</i>	517
13.3.3.	<i>Add-in AppPart with Provider Hosted applications</i>	519
13.3.4.	<i>Provider Hosted Custom Actions</i>	520

---

<b>14</b>	<b>Microsoft Teams</b>	<b>524</b>
14.1.	<i>Introduction</i>	525
14.2.	<i>Teams configuration for developing</i>	525
14.3.	<i>Developing for Teams and development tools</i>	526
14.3.1.	<i>Location of the Teams objects</i>	526
14.3.2.	<i>Teams App Studio</i>	527
14.3.3.	<i>Teams Developer Preview</i>	527
14.3.4.	<i>ngrok</i>	528
14.3.5.	<i>Cards</i>	529
14.4.	<i>Teams Tabs</i>	529
14.4.1.	<i>Personal Tabs</i>	530
14.4.2.	<i>Channel Tabs</i>	535
14.5.	<i>Bots</i>	542
14.6.	<i>Messaging Extensions</i>	550
14.6.1.	<i>Messaging Extensions with Search Commands</i>	550
14.6.2.	<i>Messaging Extensions with Action Commands</i>	557
14.7.	<i>Webhooks</i>	562
14.7.1.	<i>Incoming Webhooks</i>	562
14.7.2.	<i>Outgoing Webhooks</i>	566
14.8.	<i>SharePoint Framework (SPFx) WebParts as Teams Tabs</i>	572
14.9.	<i>Teams Tabs as SPFx WebParts for SharePoint</i>	575
14.10.	<i>Managing Teams with PowerShell</i>	576
14.10.1.	<i>Connect to Teams</i>	577
14.10.2.	<i>CRUD operations for Teams</i>	578

14.10.3.	<i>CRUD operations for Channels</i>	580
14.10.4.	<i>Users and Policies management</i>	582

---

<b>15</b>	<b>Power Automate</b>	<b>585</b>
15.1.	<i>Introduction</i>	586
15.2.	<i>Connectors</i>	586
15.2.1.	<i>Creation of a Custom Connector for Power Automate</i>	586
15.2.2.	<i>Installation of a Custom Connector for Power Automate</i>	589
15.3.	<i>Calling REST services directly</i>	590
15.3.1.	<i>Making a GET call</i>	590
15.3.2.	<i>Making a POST call</i>	591
15.4.	<i>Receiving HTTP calls directly</i>	591
15.4.1.	<i>Receiving a call with the Request action</i>	591
15.4.2.	<i>Using a Response action to react to a call</i>	592
15.5.	<i>PowerShell for Power Automate</i>	592
15.5.1.	<i>Connect to Power Automate</i>	593
15.5.2.	<i>Admin cmdlets for Power Automate</i>	594
15.5.3.	<i>Maker cmdlets for Power Automate</i>	598
15.5.4.	<i>Power Automate in SharePoint and PowerShell</i>	604

---

<b>16</b>	<b>Power Apps</b>	<b>605</b>
16.1.	<i>Introduction</i>	606
16.2.	<i>Connectors</i>	606
16.3.	<i>Connecting Power Apps with Power Automate</i>	607
16.4.	<i>PowerShell for Power Apps</i>	609
16.4.1.	<i>Connect to Power Apps</i>	609
16.4.2.	<i>Admin cmdlets for Power Apps</i>	611
16.4.3.	<i>Maker cmdlets for Power Apps</i>	615



*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.

	<p><b>EWS is becoming deprecated:</b> Microsoft has announced (<a href="https://developer.microsoft.com/en-us/graph/blogs/upcoming-changes-to-exchange-web-services-ews-api-for-office-365/">https://developer.microsoft.com/en-us/graph/blogs/upcoming-changes-to-exchange-web-services-ews-api-for-office-365/</a>) 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."</p>
---	--

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.

	<p>Regarding PowerShell, there is a set of cmdlets dedicated to <b>Exchange Online Protection</b> (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.</p>
---	---

## 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="exAppld" 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="exAppld" 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.

	<p><b>EWS Basic Authentication is becoming fully decommissioned:</b> Microsoft has announced (<a href="https://developer.microsoft.com/en-us/graph/blogs/upcoming-changes-to-exchange-web-services-ews-api-for-office-365/">https://developer.microsoft.com/en-us/graph/blogs/upcoming-changes-to-exchange-web-services-ews-api-for-office-365/</a>) 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.</p>
---	---

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 **RedirectUrlValidationCallback** 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
<b>Routines</b>		
<b>NuGets</b>	Microsoft.Exchange.WebServices, Microsoft.Identity.Client	
<b>Ref. DLLs</b>	Microsoft.Exchange.WebServices, Microsoft.Exchange.WebServices.Auth, Microsoft.Identity.Client	
<b>Using</b>	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, RedirectUrlValidationCallback);
    //Console.WriteLine(exService.Url);

    return exService;
}
    
```

```

static bool RedirectToUrlValidationCallback(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
<b>Routines</b>		
<b>NuGets</b>	Microsoft.Exchange.WebServices, Microsoft.Identity.Client	
<b>Ref. DLLs</b>	Microsoft.Exchange.WebServices, Microsoft.Exchange.WebServices.Auth, Microsoft.Identity.Client	
<b>Using</b>	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**.

	<p>The DLLs to work with the <b>Microsoft.Identity.Client</b> can be installed by the NuGet <b>Microsoft.Identity.Client by Microsoft</b> (<a href="https://www.nuget.org/packages/Microsoft.Identity.Client/">https://www.nuget.org/packages/Microsoft.Identity.Client/</a>) directly from Visual Studio.</p>
---	--

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
<b>Routines</b>		
<b>NuGets</b>	Microsoft.Exchange.WebServices, Microsoft.Identity.Client	
<b>Ref. DLLs</b>	Microsoft.Exchange.WebServices, Microsoft.Exchange.WebServices.Auth, Microsoft.Identity.Client	
<b>Using</b>	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
<b>Routines</b>		
<b>NuGets</b>	Microsoft.Exchange.WebServices, Microsoft.Identity.Client	
<b>Ref. DLLs</b>	Microsoft.Exchange.WebServices, Microsoft.Exchange.WebServices.Auth, Microsoft.Identity.Client	
<b>Using</b>	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
<b>Routines</b>		
<b>PS Modules</b>	GenericOAuthEWS.ps1	
<b>Other Modules</b>	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
<b>Routines</b>		
<b>PS Modules</b>	GenericOAuthEWS.ps1	
<b>Other Modules</b>	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 ([glencales@yahoo.com](mailto:glencales@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>.

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

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
<b>Routines</b>		
<b>PS Modules</b>	GenericOAuthEWS.ps1	
<b>Other Modules</b>	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
<b>Routines</b>		
<b>PS Modules</b>	GenericOAuthEWS.ps1	
<b>Other Modules</b>	Microsoft.Exchange.WebServices.dll	

```

Function ConnectPsOnlBA()
{
    [SecureString]$securePW = ConvertTo-SecureString `
        $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
<b>Routines</b>		
<b>PS Modules</b>	GenericOAuthEWS.ps1	
<b>Other Modules</b>	Microsoft.Exchange.WebServices.dll	

```

##==> Exchange Online PowerShell Basic Authorization
ConnectPsOnlBA

Get-Mailbox #Calling any cmdlet

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

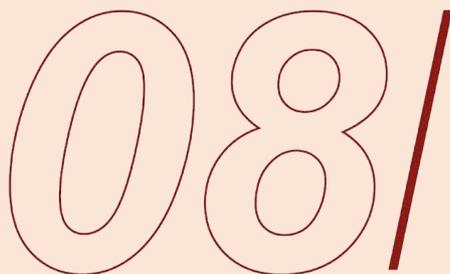
The remote session is killed at the end of the code for security concerns.

	<p>Microsoft has announced that Remote PowerShell for Exchange will be closed on October 13<sup>th</sup>, 2020 (<a href="https://techcommunity.microsoft.com/t5/blogs/blogarticleprintpage/blog-id/Exchange/article-id/27095">https://techcommunity.microsoft.com/t5/blogs/blogarticleprintpage/blog-id/Exchange/article-id/27095</a>). 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.</p>
---	---

### 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.

	<p>The EWS DLLs can be installed by the NuGet <code>Microsoft.Exchange.WebServices</code> by Microsoft (<a href="https://www.nuget.org/packages/Microsoft.Exchange.WebServices/">https://www.nuget.org/packages/Microsoft.Exchange.WebServices/</a>) directly from Visual Studio.</p>
---	---



*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#o** (for a Communication Site) will suffice. For classic Site Collections, use any of the other template identifiers.

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

```

static void SpCsCsomGetOneWebInSiteCollection()
{
    string myWebFullUrl = ConfigurationManager.AppSettings["spUrl"] +
        "/NewWebSiteModernCsSom";

    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.



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:
  - o Standard (Public) Channels that use the base SharePoint Site Collection and Exchange Group created for the Team.
  - o 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 <b>React control library</b> in App Studio will be deprecated in the future. It is recommended to use the <b>Fluent-UI react controls</b> from <a href="https://microsoft.github.io/fluent-ui-react/">https://microsoft.github.io/fluent-ui-react/</a> .
---	---

### 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](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](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/) (<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
		<pre> &lt;%@ Page Language="C#" AutoEventWireup="true" CodeBehind="GenerateGuid.aspx.cs"     Inherits="KKJA.GenerateGuid" %&gt;  &lt;!DOCTYPE html&gt;  &lt;html xmlns="http://www.w3.org/1999/xhtml"&gt; &lt;head runat="server"&gt;     &lt;title&gt;&lt;/title&gt;     &lt;script src="https://statics.teams.microsoft.com/sdk/v1.0/js/MicrosoftTeams.min.js"         type="text/javascript"&gt;&lt;/script&gt;     &lt;script src="GenerateAppScripts.js" type="text/javascript"&gt;&lt;/script&gt;     &lt;link rel="stylesheet" href="GenerateThemes.css" type="text/css" /&gt; &lt;/head&gt; &lt;body class="theme-light"&gt;     &lt;form id="form1" runat="server"&gt;         &lt;div class="surface font-semibold font-title"&gt;&lt;h2&gt;Generate a new GUID&lt;/h2&gt;&lt;/div&gt;         &lt;div&gt;             &lt;p&gt;                 &lt;asp:Button ID="btnGenerateGuid" runat="server" Text="Generate"                     OnClick="btnGenerateGuid_Click" /&gt;             &lt;/p&gt;             &lt;p class="surface"&gt;                 &lt;asp:Label ID="lblNewGuid" runat="server" Text=""&gt;&lt;/asp:Label&gt;                 &lt;asp:Label ID="lblContextInfo" runat="server" Text=""&gt;&lt;/asp:Label&gt;             &lt;/p&gt;         &lt;/div&gt;     &lt;/form&gt; </pre>

```
</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
<b>Other Modules</b>	MicrosoftTeams.min.js	
<pre>(function () {   'use strict';    microsoftTeams.initialize();    microsoftTeams.getContext(function (context) {     if (context &amp;&amp; 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);     }   } })();</pre>		



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
<b>Routines</b>		
<b>NuGets</b>		
<b>Ref. DLLs</b>		
<b>Using</b>		

```

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

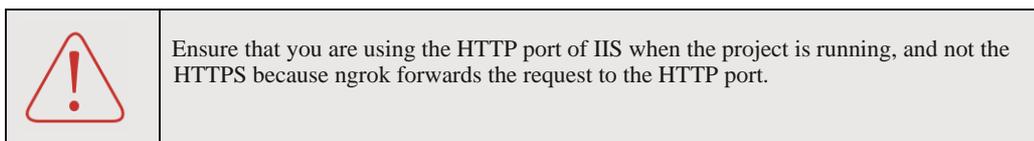
```

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
<pre> &lt;%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Privacy.aspx.cs"     Inherits="KKJA.Privacy" %&gt; &lt;!DOCTYPE html&gt; &lt;html xmlns="http://www.w3.org/1999/xhtml"&gt; &lt;head runat="server"&gt;     &lt;title&gt;&lt;/title&gt;     &lt;script src="https://statics.teams.microsoft.com/sdk/v1.0/js/MicrosoftTeams.min.js"         type="text/javascript"&gt;&lt;/script&gt;     &lt;script src="GenerateAppScripts.js" type="text/javascript"&gt;&lt;/script&gt;     &lt;link rel="stylesheet" href="GenerateThemes.css" type="text/css" /&gt; &lt;/head&gt; &lt;body class="theme-light"&gt;     &lt;form id="form1" runat="server"&gt;         &lt;div&gt;             This is the Privacy Statement page         &lt;/div&gt;     &lt;/form&gt; &lt;/body&gt; &lt;/html&gt; </pre>		

14.005	ID	File
<pre>&lt;%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Terms.aspx.cs" Inherits="KKJA.Terms" %&gt;</pre>		
<pre>&lt;!DOCTYPE html&gt;</pre>		
<pre>&lt;html xmlns="http://www.w3.org/1999/xhtml"&gt;</pre>		
<pre>&lt;head runat="server"&gt;</pre>		
<pre>&lt;title&gt;&lt;/title&gt;</pre>		
<pre>&lt;script src="https://statics.teams.microsoft.com/sdk/v1.0/js/MicrosoftTeams.min.js" type="text/javascript"&gt;&lt;/script&gt;</pre>		
<pre>&lt;script src="GenerateAppScripts.js" type="text/javascript"&gt;&lt;/script&gt;</pre>		
<pre>&lt;link rel="stylesheet" href="GenerateThemes.css" type="text/css" /&gt;</pre>		
<pre>&lt;/head&gt;</pre>		
<pre>&lt;body class="theme-light"&gt;</pre>		
<pre>&lt;form id="form1" runat="server"&gt;</pre>		
<pre>&lt;div&gt;</pre>		
<pre>    This is the Terms of Use page</pre>		
<pre>&lt;/div&gt;</pre>		
<pre>&lt;/form&gt;</pre>		
<pre>&lt;/body&gt;</pre>		
<pre>&lt;/html&gt;</pre>		

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.



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.

