

# Microsoft Power Platform Enterprise Architecture

Design tailor-made solutions for architects and decision makers to meet complex business requirements

**Second Edition**

**Robert Rybaric**



# **Microsoft Power Platform Enterprise Architecture**

Second Edition

Design tailor-made solutions for architects and decision makers to meet complex business requirements

**Robert Rybaric**



BIRMINGHAM—MUMBAI

# **Microsoft Power Platform Enterprise Architecture**

**Second Edition**

Copyright © 2023 Packt Publishing

*All rights reserved.* No part of this book may be reproduced, stored in a retrieval system, 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 Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

**Senior Publishing Product Manager:** Ashitosh Gupta

**Acquisition Editor – Peer Reviews:** Saby Dsilva

**Project Editor:** Rianna Rodrigues

**Content Development Editor:** Matthew Davies

**Copy Editor:** Safis Editing

**Technical Editor:** Aniket Shetty

**Proofreader:** Safis Editing

**Indexer:** Rekha Nair

**Presentation Designer:** Ganesh Bhadwalkar

**Developer Relations Marketing Executive:** Rohan Dobhal

First published: September 2020

Second edition: January 2023

Production reference: 1270123

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

B3 2PB, UK.

ISBN 978-1-80461-263-7

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

# Contributors

## About the author



**Robert Rybaric** is a Microsoft Power Platform and Microsoft Dynamics 365 architect, consultant, and trainer. He is a **Microsoft Certified Trainer (MCT)** and holds several certifications for Microsoft Dynamics 365, Microsoft 365, and Microsoft Azure.

For 10 years, Robert has worked for the Microsoft Corporation on numerous Microsoft Dynamics 365 presales and project implementation activities for enterprise customers across Europe, both as an architect and consultant. He is now a freelance architect, implementing Microsoft Dynamics 365 solutions for global customers and leading many Microsoft Power Platform training sessions.

In the past, Robert has written two books about Power Platform, including *Microsoft Power Platform Enterprise Architecture* (first edition) and *Microsoft Power Platform Up and Running*, published with Packt Publishing and BPB Publications respectively.

## About the reviewers

**Danilo Capuano** is a Technical Delivery Manager and Office Manager at Agic Technology. He is a Microsoft Certified Trainer (MCT) and MVP. Danilo specializes in solution architecture for Power Platform, Dynamics 365, and Azure, as well as being familiar with DevOps engineering. He is active in the Microsoft community in Italy, being a group leader for Power Apps User Group Italia and Power Pages User Group Italia. He is active on social media and can be found through his blog ([danilocapuano.blog](http://danilocapuano.blog)), Twitter (@capuanodanilo), and LinkedIn (/capuanodanilo).

He has previously worked as a technical reviewer on several other titles, including *Fundamentals of CRM with Dynamics 365 and Power Platform*, *Mastering Microsoft Dynamics NAV 2016*, *Microsoft Dynamics NAV 7 Programming Cookbook*, *Microsoft Dynamics NAV 2013 Application Design*, *Learn Microsoft PowerApps*, and *Programming Microsoft Dynamics NAV 2015*.

**EY Kalman** started as a Dynamics 365 Support Analyst over 15 years ago and is now an experienced Power Platform and Dynamics 365 Solution Architect and Microsoft MVP. He has run development teams over large-scale projects across multiple sectors, translating business ideas into technical requirements and delivering projects. He has previously worked as a technical reviewer on another title, *Microsoft Power Platform Functional Consultant: PL-200 Exam Guide*.

EY currently enjoys being a technical evangelist for adopting business application technologies, enabling and empowering proper digital transformation for enterprise users across the globe. He engages regularly with the Microsoft Business Applications community under his handle of ‘The CRM Ninja’. EY blogs regularly about technical ideas, runs a weekly show called ‘The Oops Factor’, and organizes and presents at events. He also enjoys playing with gadgets, traveling, and going out for rides on his Ninja motorbike.

# Table of Contents

<b>Preface</b>	<b>xxvii</b>
<hr/>	
<b>Section I: The Basics</b>	<b>1</b>
<hr/>	
<b>Chapter 1: Microsoft Power Platform and Microsoft Dynamics 365 Overview</b>	<b>3</b>
<hr/>	
Introducing Contoso Inc. .....	4
Introducing Microsoft Power Platform .....	4
Introducing the Common Data Model and Microsoft Dataverse • 6	
<i>Introducing the Common Data Model</i> • 6	
<i>Introducing Microsoft Dataverse</i> • 7	
Introducing model-driven apps • 7	
Introducing canvas apps • 9	
Introducing Power Automate • 11	
Introducing Power Virtual Agents • 12	
Introducing Power BI • 12	
Introducing On-Premises Data Gateway • 13	
Introducing AI Builder • 13	
Introducing Power Pages • 14	
<b>Introducing Microsoft Dynamics 365 CRM applications</b> .....	<b>15</b>
Microsoft Dynamics 365 Sales • 15	
Microsoft Dynamics 365 Marketing • 15	
Microsoft Dynamics 365 Customer Service • 16	

---

Microsoft Dynamics 365 Field Service • 16	
Microsoft Dynamics 365 Project Operations • 17	
<b>Introducing Microsoft Dynamics 365 ERP applications .....</b>	<b>17</b>
Microsoft Dynamics 365 Finance • 17	
Microsoft Dynamics 365 Supply Chain Management • 18	
Microsoft Dynamics 365 Commerce • 18	
Microsoft Dynamics 365 Human Resources • 19	
Microsoft Dynamics 365 Business Central • 19	
<b>Introducing Microsoft Dynamics 365 AI, MR, and other modules .....</b>	<b>19</b>
Microsoft Dynamics 365 Customer Insights • 20	
Microsoft Dynamics 365 Sales Insights • 20	
Microsoft Dynamics 365 Connected Spaces • 21	
Microsoft Dynamics 365 Fraud Protection • 21	
Microsoft Dynamics 365 Remote Assist • 21	
Microsoft Dynamics 365 Guides • 21	
Microsoft Dynamics 365 Product Visualize • 21	
Microsoft Dynamics 365 Unified Service Desk • 22	
<b>Microsoft Power Platform licensing overview .....</b>	<b>22</b>
<b>Contoso Inc. Power Platform commitment .....</b>	<b>24</b>
<b>Summary .....</b>	<b>25</b>
<b>Chapter 2: Microsoft 365 and Microsoft Azure Overview</b>	<b>27</b>
<b>Contoso Inc. cloud maturity .....</b>	<b>27</b>
<b>Introducing Microsoft 365 .....</b>	<b>28</b>
Introducing Microsoft Office 365 • 29	
<i>Microsoft Exchange</i> • 30	
<i>Microsoft SharePoint</i> • 30	
<i>Microsoft OneDrive</i> • 30	
<i>Microsoft OneNote</i> • 30	

<i>Microsoft Teams</i> • 30	
<i>Microsoft Outlook</i> • 30	
<i>Microsoft Word</i> • 31	
<i>Microsoft Excel</i> • 31	
Overviewing Microsoft Enterprise Mobility + Security • 31	
<i>Microsoft Intune</i> • 31	
<i>Microsoft Cloud App Security</i> • 31	
<i>System Center Configuration Manager</i> • 32	
Contoso Inc. using Microsoft 365 • 32	
<b>Introducing Microsoft Azure .....</b>	<b>33</b>
Introducing Azure Active Directory • 34	
Introducing Azure Service Bus • 34	
Introducing Azure Event Hubs • 34	
Introducing Azure Logic Apps • 35	
Introducing Azure API Management • 35	
Introducing Azure Functions • 35	
Introducing Azure SQL • 35	
Introducing Azure Cosmos DB • 36	
Introducing Azure Blob Storage • 36	
Introducing Azure Synapse Analytics • 36	
Introducing Azure IoT Hub and Azure IoT Central • 37	
Introducing Azure Key Vault • 37	
Introducing Azure DevOps • 37	
Introducing Azure Monitor • 37	
Contoso Inc. using Microsoft Azure • 38	
<b>Microsoft 365 and Microsoft Azure licensing overview .....</b>	<b>39</b>
Microsoft 365 licensing • 39	
Microsoft Azure licensing • 39	
<b>Summary .....</b>	<b>40</b>

<b>Section II: The Architecture</b>	<b>43</b>
<hr/>	
<b>Chapter 3: Understanding the Microsoft Power Platform Architecture</b>	<b>45</b>
<hr/>	
Contoso Inc. starts architecting their planned Power Platform solution .....	46
Understanding the Power Platform architecture .....	46
Learning about the Microsoft cloud infrastructure • 46	
Understanding the customer cloud structure • 47	
<i>User management</i> • 49	
<i>License management</i> • 49	
<i>Group management</i> • 49	
<i>App registration</i> • 49	
<i>Office 365 Activity Logging</i> • 49	
Learning about Power Platform technology • 50	
Understanding Power Platform environments • 50	
<i>Microsoft Dataverse</i> • 53	
<i>Microsoft Dataverse for Teams</i> • 53	
<i>Capacity restrictions</i> • 53	
<i>Power Platform data connectors</i> • 56	
<i>Data Loss Prevention policies</i> • 57	
<i>On-Premises Data Gateway</i> • 58	
<i>Managed environments</i> • 60	
Learning about Power BI's structure • 60	
Understanding the Power Platform and Dynamics 365 clients .....	62
Learning about desktop clients • 62	
<i>Browser client</i> • 62	
<i>Dynamics 365 App for Outlook</i> • 63	
<i>Unified Service Desk</i> • 63	
<i>Omnichannel for Dynamics 365 Customer Service</i> • 65	
<i>Robotic process automation with Power Automate Desktop flows</i> • 65	
Understanding mobile clients • 66	

<b>Learning about Power Platform administration and monitoring .....</b>	<b>67</b>
Understanding Power Platform administration centers •	67
<i>Microsoft Azure portal</i> •	67
<i>Microsoft 365 Admin Center</i> •	68
<i>Power Platform Admin Center</i> •	68
<i>Power BI admin portal</i> •	70
Understanding PowerShell administration and monitoring •	71
<i>Microsoft 365 administration</i> •	71
<i>Power Platform administration</i> •	72
<i>Power BI administration</i> •	73
<i>PowerShell monitoring</i> •	74
Learning about API administration •	74
<i>Microsoft 365 administration</i> •	75
<i>Power Platform administration</i> •	76
<i>Power BI administration</i> •	76
Administration and monitoring using Power Automate •	78
Administration using Azure DevOps •	80
Learning about platform auditing •	80
<i>Office 365 Activity Logging</i> •	81
<i>Dataverse auditing</i> •	82
Understanding application monitoring •	82
<b>Presenting architectural best practices .....</b>	<b>83</b>
Introducing single tenants or multiple tenants •	83
<i>Development and testing environments</i> •	83
<i>Unsupported integration topology</i> •	84
Understanding environment strategies •	87
<i>Default environment</i> •	88
<i>Developer environment</i> •	88
<i>Shared test and production environment</i> •	88
<i>Dedicated environment</i> •	90
<i>Complex testing</i> •	91

<i>Multiple release strategy</i> • 91	
<i>Product upgrades</i> • 92	
<i>Other environment types</i> • 94	
Environment regions • 94	
Administration and monitoring • 95	
<b>Contoso Inc. Power Platform architecture .....</b>	<b>97</b>
Tenant structure • 98	
Power Platform environments • 99	
Power Platform clients • 100	
User groups and licensing • 101	
<b>Summary .....</b>	<b>101</b>
<hr/>	
<b>Chapter 4: Power Platform Customization and Development Tools and Techniques</b>	<b>103</b>
<hr/>	
<b>Contoso Inc. empowering the project team .....</b>	<b>103</b>
<b>The citizen developer vs. IT pro developer paradigm .....</b>	<b>104</b>
Introducing the citizen developer • 104	
The IT pro developer • 104	
Distinguishing between the citizen developer and the IT pro developer • 105	
<b>Presenting configuration and customization tools .....</b>	<b>105</b>
Microsoft Dataverse and model-driven app tools • 105	
<i>Power Apps Maker Portal</i> • 106	
<i>Model-driven app designer</i> • 107	
<i>Introducing XrmToolBox</i> • 108	
Power Apps Studio • 109	
Power Pages Studio • 111	
Power Automate Maker Portal • 112	
Power Virtual Agents designer • 113	
AI Builder • 113	
Dataflows designer • 115	
Power BI designer tools • 115	

<i>Power BI Desktop</i> • 116	
<i>Power BI Builder</i> • 117	
<i>Power BI service</i> • 117	
Microsoft AppSource • 118	
ISV Studio • 118	
<b>Presenting custom development tools .....</b>	<b>118</b>
Visual Studio • 118	
Visual Studio Code • 119	
Power Apps Command-Line Interface (CLI) • 119	
Power Platform extensions for Visual Studio • 120	
NuGet developer tools and assemblies • 120	
<i>The code generation tool</i> • 120	
<i>The plug-in registration tool</i> • 121	
XrmToolBox • 121	
Postman • 121	
CRMRestBuilder • 121	
Testing tools • 122	
<i>Testing the user interface</i> • 122	
<i>Testing backend components</i> • 122	
<i>Using network traffic analyzers</i> • 122	
<b>Presenting application lifecycle management tools .....</b>	<b>123</b>
NuGet developer tools and assemblies • 123	
<i>The configuration migration tool</i> • 123	
<i>The package deployer tool</i> • 123	
<i>The solution packager tool</i> • 124	
Azure DevOps • 124	
<b>Contoso Inc. project team workplace setup .....</b>	<b>125</b>
Enabling the core project team • 125	
Enabling citizen developers • 126	
<b>Summary .....</b>	<b>126</b>

<b>Chapter 5: Application Lifecycle Management</b>	<b>127</b>
Contoso Inc. implementing application lifecycle management .....	127
Understanding application lifecycle management .....	128
Environment complexity • 128	
Power Platform solution complexity • 129	
ALM for the Power Platform • 130	
Introducing solutions management .....	131
Overview of solutions • 131	
<i>Environment variables</i> • 132	
<i>Solution properties</i> • 133	
Solution types • 134	
<i>Unmanaged solution</i> • 134	
<i>Managed solution</i> • 135	
<i>Default solution</i> • 135	
<i>Common Data Services Default Solution</i> • 135	
Managed properties • 136	
<i>Forms, views, charts, and dashboards properties</i> • 136	
<i>Columns properties</i> • 136	
<i>Tables properties</i> • 136	
Dependencies and solution segmentation • 137	
<i>Solution layering</i> • 137	
<i>Layering behavior</i> • 138	
<i>Solution dependencies</i> • 139	
<i>Solution segmentation</i> • 139	
Patching and updating solutions • 140	
<i>Solution patch</i> • 140	
<i>Solution updates</i> • 140	
Microsoft updates • 142	
<i>Major Power Platform updates</i> • 142	
<i>First-party applications updates</i> • 142	
<i>Power Platform hotfixes</i> • 143	

---

<b>Introducing Azure DevOps for the Power Platform .....</b>	<b>143</b>
Overview of Power Platform Build Tools • 143	
<i>Helper and quality check tasks</i> • 144	
<i>Solution tasks</i> • 144	
<i>Environment tasks</i> • 145	
Azure DevOps with Power Platform Build Tools • 145	
<i>Committing a solution to the source control</i> • 146	
<i>Distributing solutions between development environments</i> • 146	
<i>Distributing solutions out of development</i> • 148	
<i>Pipeline versus Release</i> • 150	
<b>Introducing GitHub for the Power Platform .....</b>	<b>150</b>
<b>Application lifecycle management for Power BI .....</b>	<b>151</b>
Environments in Power BI • 152	
Power BI components • 153	
Power BI ALM approach • 153	
Power BI deployment pipelines • 154	
<b>Application lifecycle management for other solution components .....</b>	<b>155</b>
<b>Application lifecycle management best practices .....</b>	<b>155</b>
Solution best practices • 155	
<i>General practices</i> • 155	
<i>Unmanaged versus managed</i> • 156	
<i>Structuring solutions</i> • 156	
<i>Using segmentation</i> • 158	
<i>Using source control</i> • 159	
Solution publishers' best practices • 159	
Power BI best practices • 160	
<b>Contoso Inc. ALM strategy .....</b>	<b>160</b>
Establishing Azure DevOps • 160	
Using Power Platform solutions • 161	
Using Power BI ALM • 161	
Other ALM decisions • 162	
<b>Summary .....</b>	<b>162</b>

<b>Section III: The Implementation</b>	<b>165</b>
<b>Chapter 6: Implementation Approach and Methodologies</b>	<b>167</b>
Contoso Inc. preparing the implementation project .....	167
Getting an overview of the implementation approach .....	168
Understanding customer enterprise architecture and environment .....	169
Data residency requirements • 170	
Authentication providers • 170	
Internet restrictions • 170	
Data protection requirements • 171	
Learning about project implementation methodologies and tools .....	171
Understanding programs and projects • 171	
Understanding project implementation methodologies • 172	
<i>The waterfall model</i> • 173	
<i>The agile model</i> • 174	
<i>The iterative model</i> • 175	
<i>The combined model</i> • 176	
Making a project effort estimation • 177	
<i>Business requirements</i> • 177	
<i>Custom development</i> • 178	
<i>Infrastructure requirements</i> • 178	
<i>Integration</i> • 178	
<i>Data migration</i> • 179	
<i>Other efforts</i> • 180	
Project management tools • 180	
<i>Microsoft Project</i> • 180	
<i>Azure DevOps</i> • 181	
<i>Effort estimators</i> • 181	
Creating project documentation • 182	
<i>The project plan</i> • 182	

<i>The requirements document</i> • 184	
<i>The solution architecture document</i> • 184	
<i>The solution/technical design document</i> • 185	
<i>Other documents</i> • 185	
<b>Learning about project setup .....</b>	<b>186</b>
Project types • 186	
<i>Internal project</i> • 186	
<i>External project</i> • 187	
Project roles and responsibilities • 188	
<i>Central roles and responsibilities</i> • 189	
<i>Partner roles and responsibilities</i> • 190	
<i>Customer roles and responsibilities</i> • 194	
<b>Understanding project phases .....</b>	<b>198</b>
The preparation phase • 198	
<i>Identifying demand</i> • 198	
<i>Studying feasibility</i> • 199	
<i>Specifying the budget</i> • 200	
<i>Seeking approval</i> • 200	
<i>Issuing a Request for Information (RFI)</i> • 200	
<i>Issuing an RFP/RFQ/RFT</i> • 201	
<i>Discovery</i> • 203	
<i>Negotiations</i> • 203	
<i>Contract</i> • 203	
The project execution phase • 203	
<i>Project preparation</i> • 204	
<i>Project initiation</i> • 204	
<i>Initial analysis</i> • 204	
<i>Iterative execution</i> • 205	
<i>Final testing</i> • 208	
<i>Solution deployment</i> • 209	
The operation phase • 209	

---

<i>Support transition</i> • 210	
<i>Operation</i> • 210	
<i>Decommission</i> • 211	
<b>Contoso Inc. starting the implementation project .....</b>	<b>211</b>
<i>Bidding process</i> • 212	
<i>Project setup and methodology</i> • 212	
<i>Project plan, tools, and documentation</i> • 213	
<i>Project setup</i> • 214	
<b>Summary .....</b>	<b>215</b>
<b>Chapter 7: Microsoft Power Platform Security</b>	<b>217</b>
<b>Contoso Inc. designing Power Platform solution security .....</b>	<b>217</b>
<b>Getting an overview of IT security .....</b>	<b>218</b>
<i>Authentication versus authorization</i> • 218	
<i>Microsoft cloud authentication and authorization fundamentals</i> • 219	
<i>Provisioning user identity</i> • 220	
<i>Assigning licenses</i> • 221	
<i>Granting authorization</i> • 221	
<b>Understanding authentication .....</b>	<b>221</b>
<i>Identity and authentication solutions for internal users</i> • 221	
<i>Cloud identity approach</i> • 222	
<i>Password hash synchronization approach</i> • 223	
<i>Pass-through authentication approach</i> • 224	
<i>Federation approach</i> • 226	
<i>Conclusion</i> • 227	
<i>Authentication features for internal users</i> • 228	
<i>Conditional access</i> • 228	
<i>ADFS claim rules</i> • 229	
<i>Multi-factor authentication</i> • 229	
<i>Single sign-on options</i> • 230	
<i>Cross-tenant inbound and outbound restrictions</i> • 230	

Service authentication for internal users • 231	
<i>Dataverse authentication</i> • 231	
<i>Power BI authentication</i> • 235	
Authentication governance for internal users • 235	
<i>Dataverse user accounts provisioning governance</i> • 235	
<i>Dataverse session governance</i> • 236	
Azure Active Directory guest users • 237	
Authenticating external users • 237	
<b>Understanding authorization .....</b>	<b>239</b>
Authorization in Power Platform • 239	
Authorization in Dataverse and model-driven apps • 241	
<i>Fundamentals of Dataverse authorization</i> • 241	
<i>Setting up basic authorization</i> • 246	
<i>Standard role-based security</i> • 247	
<i>Modernized Business Units</i> • 248	
<i>Group teams</i> • 249	
<i>Authorizing model-driven app access</i> • 250	
<i>Hierarchy security</i> • 252	
<i>Record sharing</i> • 253	
<i>Access teams</i> • 253	
<i>Column-level security</i> • 254	
<i>User interface security</i> • 254	
Authorization in canvas apps • 255	
<i>Authorization of apps</i> • 255	
<i>Authorization of connections</i> • 255	
Authorization in Power Automate • 256	
<i>Authorization of background flows</i> • 256	
<i>Authorization of interactive flows</i> • 257	
<i>Authorization of desktop flows</i> • 257	
Authorization in Power BI • 257	
Authorization in Power Pages • 258	

Understanding compliance, privacy, and data protection .....	259
Presenting security best practices .....	261
Dataverse security roles • 262	
<i>Modifying security roles</i> • 262	
<i>Layering of security roles</i> • 262	
Dataverse content-based security • 263	
<i>Using business units</i> • 264	
<i>Using table form switching</i> • 264	
<i>Using client-side scripting or business rules</i> • 264	
<i>Using server-side event handlers</i> • 264	
Integrate security across solution components • 265	
<i>Dataverse-SharePoint integrated security</i> • 266	
<i>Dataverse-Power BI integrated security</i> • 267	
Using identity and access management automation • 267	
Establishing the Power Platform mature security model • 269	
<b>Contoso Inc. security architecture .....</b>	<b>271</b>
Active Directory integration • 271	
Data Loss Prevention policies • 272	
Dataverse • 272	
Other security decisions • 272	
<b>Summary .....</b>	<b>273</b>
<hr/> <b>Chapter 8: Microsoft Power Platform Extensibility</b>	<b>275</b>
Contoso Inc. – designing the Power Platform solution .....	276
Getting an overview of extensibility .....	276
Presenting Dataverse and model-driven app extensibility .....	278
Dataverse standard customization • 278	
<i>Dataverse data modeling</i> • 280	
<i>Dataverse user interface design</i> • 284	
<i>Designing model-driven applications</i> • 291	
<i>Designing mobile apps</i> • 292	

Dataverse automation • 292	
<i>Dataverse business rules</i> • 292	
<i>Classic Dataverse workflows</i> • 293	
<i>Dataverse custom actions</i> • 294	
<i>Dataverse custom APIs</i> • 295	
<i>Dataverse business process flows</i> • 296	
Dataverse client-side extensibility • 297	
<i>Standard custom controls</i> • 297	
<i>Power Apps Component Framework</i> • 299	
<i>Web resources</i> • 301	
<i>Embedding canvas apps</i> • 306	
Dataverse server-side extensibility • 308	
<i>Dataverse API interface types</i> • 308	
<i>Plug-in event handlers</i> • 311	
<i>Custom workflow actions</i> • 313	
<i>Azure Service Bus integration</i> • 313	
<i>Azure Event Hub integration</i> • 316	
<i>Webhook integration</i> • 316	
<i>Building external applications</i> • 317	
Unified Service Desk extensibility • 319	
<b>Presenting Power Pages extensibility .....</b>	<b>320</b>
<b>Presenting Power Automate flows .....</b>	<b>321</b>
Cloud flows • 321	
Desktop flows • 323	
Process advisor • 325	
<b>Presenting canvas apps and Power Automate extensibility .....</b>	<b>326</b>
Canvas apps and Power Automate customization • 326	
Power Fx • 327	
Building custom connectors • 328	
<b>Presenting Power BI extensibility .....</b>	<b>329</b>
<b>Power Platform extensibility best practices .....</b>	<b>330</b>

Dataverse client-side interface extensibility • 330	
Dataverse server-side extensibility • 331	
<i>Dataverse API selection</i> • 331	
<i>Extensibility and automation options</i> • 332	
<i>Performance impact</i> • 334	
<b>Contoso Inc. Power Platform solution design .....</b>	<b>334</b>
Model-driven apps • 334	
Automations • 335	
Client-side extensibility • 335	
Server-side extensibility and integrations • 336	
Other design decisions • 336	
Summary .....	336
Further reading .....	337
<b>Chapter 9: Microsoft Power Platform Integration</b>	<b>339</b>
<b>Contoso Inc. designing the Power Platform integration .....</b>	<b>340</b>
<b>Getting an overview of Power Platform integration .....</b>	<b>340</b>
<b>Integrating with Microsoft 365 and Microsoft Azure .....</b>	<b>341</b>
Introducing implicit Dynamics 365 integrations • 341	
Integrations with Microsoft 365 services • 342	
<i>Integrating with Exchange</i> • 342	
<i>Integrating with SharePoint</i> • 344	
<i>Integrating with OneDrive</i> • 345	
<i>Integrating with Microsoft Teams</i> • 346	
<i>Integrating with Microsoft OneNote</i> • 348	
<i>Integrating with Skype/Skype for Business</i> • 349	
Integrations with Microsoft Azure services • 351	
<i>Integrating with Azure Blob Storage</i> • 351	
<i>Integrating with Azure Data Lake/Synapse Analytics</i> • 352	
<i>Advanced integration scenarios with the Azure Synapse Link for Dataverse</i> • 355	
<i>Integrating with Azure Logic Apps</i> • 358	

<i>Integrating with Azure API Management</i> • 359	
<b>Frontend integration patterns and solution approaches .....</b>	<b>360</b>
Embedding third-party content into Dataverse • 360	
Embedding Dataverse content into third-party containers • 362	
Event-driven and on-demand frontend integration • 363	
Using Unified Service Desk • 364	
<b>Backend integration patterns and solution approaches .....</b>	<b>364</b>
The remote procedure call pattern • 364	
The relay pattern • 367	
The publish-subscribe pattern • 370	
The request-callback pattern • 374	
Data integration • 376	
<i>Virtual tables</i> • 377	
<i>Standard Azure data integrations</i> • 380	
<i>Integration between Dynamics 365 CRM and ERP</i> • 380	
Custom backend integrations • 381	
<b>Other Power Platform integrations .....</b>	<b>381</b>
Power Virtual Agent and AI Builder • 381	
Power BI • 383	
<b>Learning about Power Platform integration best practices .....</b>	<b>384</b>
Frontend integration • 384	
Backend integration • 384	
<b>Contoso Inc. Power Platform integration design .....</b>	<b>386</b>
Integration with Microsoft 365 and Microsoft Azure • 386	
Frontend integration • 386	
Backend integration • 387	
<b>Summary .....</b>	<b>387</b>
<hr/> <b>Chapter 10: Microsoft Power Platform Data Migration</b>	<b>389</b>
<b>Contoso Inc. planning the data migration .....</b>	<b>390</b>
<b>Getting an overview of data migration .....</b>	<b>390</b>

Migration as part of integration • 390	
Migrating consolidated data • 392	
Advanced migration • 392	
<b>Understanding the data migration tools and techniques .....</b>	<b>394</b>
Entering data manually • 394	
Using Excel files • 394	
Using the Dataverse data import wizard • 397	
The Configuration Migration Tool • 398	
Dataflows with Power Query • 400	
SQL Server Integration Services • 402	
<i>Extracting data</i> • 402	
<i>Transforming data</i> • 405	
<i>Loading data</i> • 406	
Azure Data Factory • 406	
Data migration using code • 408	
<b>Data migration challenges and best practices .....</b>	<b>409</b>
Planning and effort estimation • 409	
Scoping the migration • 409	
Understanding the impact on storage • 410	
Compliance considerations • 410	
Understanding access issues • 410	
Coping with a lack of knowledge • 411	
Dealing with a lack of documentation • 411	
Poor-quality source data • 411	
Understanding encoding issues • 412	
Understanding record ownership issues • 412	
Understanding mapping issues • 412	
Understanding record relationship issues • 413	
Understanding business process flow issues • 413	
Understanding record status issues • 414	
Setting certain system fields • 415	

---

Migrating documents • 415	
Understanding the importance of the order of migration steps • 415	
Understanding migration automation • 416	
Understanding migration performance • 417	
Resolving API limits • 417	
Time for migration execution • 418	
Verifying the data by the customer • 418	
Contoso Inc. data migration design .....	419
Summary .....	422
<b>Appendix</b>	<b>425</b>
<b>Best Practices for Solution Architecture</b>	<b>427</b>
Architectural best practices .....	428
When to use multiple tenants • 428	
<i>Separate development and testing environments</i> • 428	
<i>Unsupported integration topology</i> • 429	
Environment strategies for an enterprise-scale project • 432	
<i>The shared test and production environment strategy</i> • 433	
<i>The dedicated environments strategy</i> • 434	
<i>The complex testing strategy</i> • 435	
<i>The multiple release strategy</i> • 436	
<i>The product upgrades strategy</i> • 437	
Environment regions • 438	
Administration and monitoring • 438	
Application lifecycle management best practices .....	440
Create a specific solution package • 441	
When to use unmanaged or managed solutions • 441	
When to use a single solution • 441	
When to use multiple solutions • 441	
Component sharing and component libraries • 444	

<i>Using segmentation</i> • 444	
<i>Using source control</i> • 445	
Use one single publisher for all solutions within a project • 445	
Power BI best practices • 445	
<b>Security best practices</b> .....	<b>446</b>
Dataverse security roles • 446	
Create new custom roles instead of modifying default roles • 446	
<i>Layer security roles instead of configuring individual roles</i> • 446	
Dataverse content-based security • 448	
<i>Using a business unit hierarchy</i> • 448	
<i>Using table form switching</i> • 449	
<i>Using server-side event handlers</i> • 449	
Integrate security across solution components • 450	
<i>Dataverse-SharePoint integrated security</i> • 451	
<i>Dataverse-Power BI integrated security</i> • 452	
How to use identity and access management automation • 452	
Establishing a Power Platform mature security model • 454	
<b>Extensibility best practices</b> .....	<b>456</b>
Optimizing the performance of client-side extensibility • 456	
Dataverse server-side extensibility • 457	
<i>When to use which Dataverse API</i> • 458	
<i>Recapping extensibility and automation options</i> • 459	
<i>Avoid synchronous workflows and plug-ins</i> • 460	
<b>Integration best practices</b> .....	<b>460</b>
Frontend integration • 461	
Backend integration • 461	
<b>Data migration best practices</b> .....	<b>462</b>
Don't underestimate the project duration • 462	
Determine the scope of the migration • 462	
Understand the impact on storage • 463	
Compliance considerations • 464	

Start getting physical access to all required systems and solutions early • 464	
Expect a lack of knowledge about legacy IT systems • 464	
Include contractual responsibility for the quality of source data • 464	
Legacy IT systems might have encoding issues • 465	
Attempt to resolve record ownership issues in the data transformation phase • 465	
Understanding mapping issues • 466	
Exclude records with record relationship issues • 467	
Understanding business process flow issues • 467	
Understanding record status issues • 468	
Ascertain whether you need to set certain system fields • 468	
Migrating documents • 469	
Follow the order of migration steps • 469	
Understanding migration automation • 470	
Understanding migration performance • 470	
Request the lifting of API limits for large projects • 471	
Arrange time for the migration execution • 471	
The customer should verify the quality of all migrated data before the final migration • 472	

---

<b>Other Books You May Enjoy</b>	<b>475</b>
----------------------------------	------------

---

<b>Index</b>	<b>479</b>
--------------	------------



# Preface

Microsoft Power Platform is a very popular collection of cloud services for building business applications quickly, efficiently, and using the low-code/no-code approach. There are a huge amount of possibilities when using this platform. You can use Power Apps to build business applications for PC or mobile devices. You can build publicly available portals with Power Pages. With Power Automate, you can create cross-application automation and integration solutions. Power BI is a well-established and flexible enterprise analytical and reporting tool. Power Virtual Agents can be used to quickly build and integrate chatbots. And that's just a brief overview.

In this book, you will learn about the various components of the Power Platform from an architect's point of view. You will learn how to architect and design complex Power Platform solutions for large and global organizations. You will dive deep into Power Platform security, extensibility, integration, and data migration. This knowledge will cumulate into setting up an enterprise implementation project and successfully implement a Power Platform solution for a large organization.

While reading the book, you will accompany a fictitious global corporation called *Contoso Inc.* on their journey into the Power Platform cloud. You will see how they adopt the products, prepare the implementation project, learn all the aspects of a complex solution, and take decisions in line with best practices, all while following internal policies and rules.

After reading this book, you should have full clarity about the Power Platform technology and how to use it to deliver a robust, scalable, and user-friendly solution to your clients.

## Who this book is for

This book is intended for solution and technical architects who would like to understand all the aspects of a successful Power Platform project implementation at a high level. But of course, every technical specialist on the project implementation side and on the client side will benefit from reading this book, if they are willing to learn the details of the Power Platform.

This is not a beginner's book. Basic knowledge of Microsoft technology generally and the Power Platform in particular is required to understand the presented concepts. It is expected that the reader has basic knowledge about concepts such as PowerShell, programming languages including C# or JavaScript, and understands principles like data modeling, web services, and user interface design.

## What this book covers

### Section 1: The Basics

*Chapter 1, Microsoft Power Platform and Microsoft Dynamics 365 Overview*, gives you an overall look at the components of Microsoft Power Platform. You will establish a firm understanding of Microsoft Dynamics 365 modules for building CRM and ERP solutions as well as the other modules for leveraging artificial intelligence and using HoloLens. At the end, you will learn the basics of Power Platform and Dynamics 365 licensing.

In *Chapter 2, Microsoft 365 and Microsoft Azure Overview*, you will get an overview of the two other Microsoft cloud services, Microsoft 365 and Microsoft Azure, in the context of a Power Platform solution. You will learn about all the components used in this context, and how a solution can benefit from integrating with them. You will also get a basic overview of how those cloud components are licensed.

### Section 2: The Architecture

In *Chapter 3, Understanding the Microsoft Power Platform Architecture*, we will dive deep into the world of Power Platform architecture. You will learn about the infrastructure, architecture, and structure of the Power Platform cloud components. You will see what client components are available for the various devices used today. After that, we will focus on the administration and monitoring possibilities before turning to the various architectural best practices proven for Power Platform implementations in large organizations.

*Chapter 4, Power Platform Customization and Development Tools and Techniques*, will cover the tools and techniques used to configure and customize Power Platform solutions. These tools and techniques are for when the low-code/no-code approach does not cover advanced client requirements and custom development is necessary. You will also learn about some of the tools that support the application lifecycle of Power Platform solutions.

In *Chapter 5, Application Lifecycle Management*, you will become familiar with the possibilities that open up when adopting **application lifecycle management (ALM)** principles for Power Platform solution architecture. You will learn the details about solution management as the main ALM approach for all parts of a solution, aside from Power BI. Then, we will have a look at the possibilities for ALM in Power BI. You will also understand how Microsoft Azure DevOps and GitHub can help you make ALM smooth, easy, and fully automated. At the end, you will be presented with a collection of ALM related best practices.

## Section 3: The Implementation

*Chapter 6, Implementation Approach and Methodologies*, is a specific chapter where you will learn a lot of practical details on how to understand client's enterprise architecture. You will see what project implementation methodologies are available and often used for Power Platform projects and how to prepare an implementation project. Finally, we will look at a typical project setup in terms of roles and responsibilities and an enterprise project lifecycle from the very beginning to when the solution is brought into production.

In *Chapter 7, Microsoft Power Platform Security*, we will focus on all aspects of Power Platform solution security. You will learn all the details of authentication within Microsoft cloud solutions with specifics for Power Platform. Next you will see how the authorization in the various Power Platform components can be implemented. At the end, you will learn a large array of security-related best practices.

In *Chapter 8, Microsoft Power Platform Extensibility*, you will dive deep into the extensibility of various Power Platform components. You will learn what can be achieved by configuration and customization as well as what types of requirements need to be developed with code. This chapter is mainly dedicated to Dataverse applications, but you will also gain an understanding of the extensibility of canvas apps, Power Automate, and Power BI. As usual, at the end, we present various extensibility best practices.

*Chapter 9, Microsoft Power Platform Integration*, will explain to you that every complex Power Platform solution is heavily integrated with two things: other Microsoft cloud services and the client's own IT ecosystem. That's why it is very important to understand the integration possibilities for Power Platform. In this chapter, you will learn about all those Microsoft 365 and Microsoft Azure integration options, from which most can be achieved by a simple configuration. Further, you will see how a custom frontend and backed integration can be implemented and what the typical integration patterns and solution approaches are. Additionally, we will explore several integration capabilities of Power Virtual Agents, AI Builder, and Power BI.

*Chapter 10, Microsoft Power Platform Data Migration*, is dedicated to data migration. In most cases, the successful implementation of a Power Platform solution must be accompanied with a data migration effort to bring all the useful data from a client's various legacy IT systems and solutions into Power Platform. In this chapter, you will learn the usual data migration strategies and what tools and solutions can be used for this purpose. At the end you will be presented with best practices to mitigate the different challenges of a complex data migration.

## Appendix

*Best Practices for Solution Architecture*, found at the end of this book, consolidates the key principles of each chapter into one place. You will find professional advice regarding architecture, ALM, security, and extensibility. Along the way, you will revisit topics and resolve dilemmas, such as evaluating environment strategies or identifying the scenarios in which you would need multiple tenants. Finally, there will be a list of the best practices to follow when performing data migration.

## To get the most out of this book

This book is dedicated to Microsoft Power Platform, so a permanent or temporary (trial) license of the Power Platform cloud components is required. Further, there is a number of tools necessary to perform configuration, customization, and custom development on the Power Platform. Those tools are described in detail in *Chapter 4, Power Platform Customization and Development Tools and Techniques*.

## Download the example code files

The code bundle for the book is hosted on GitHub at <https://github.com/PacktPublishing/Microsoft-Power-Platform-Enterprise-Architecture-2E>. We also have other code bundles from our rich catalog of books and videos available at <https://github.com/PacktPublishing/>. Check them out!

## Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: <https://packt.link/CTyMi>.

## Conventions used

There are a number of text conventions used throughout this book.

**CodeInText:** Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. For example: “The authentication process is using the URL, `clientId`, `clientsecret`, and the ID of the AAD tenant to perform the authentication process.”

A block of code is set as follows:

```
string url = "https://contoso.crm.dynamics.com";
string clientId = "51f81489-12ee-4a9e-aaaе-a2591f45987d";
string clientsecret = "<yourclientsecret>";
string tenantid = "<yourtenantid>";
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
string url = "https://contoso.crm.dynamics.com";
string clientId = "51f81489-12ee-4a9e-aaaе-a2591f45987d";
string clientsecret = "<yourclientsecret>";
string tenantid = "<yourtenantid>";
```

Any command-line input or output is written as follows:

```
Get-AdminPowerAppEnvironment *Contoso*
```

**Bold:** Indicates a new term, an important word, or words that you see on the screen. For instance, words in menus or dialog boxes appear in the text like this. For example: “**Model-driven apps** integration with **Application Insights** needs to be implemented with code.”



Warnings or important notes appear like this.



Tips and tricks appear like this.

## Get in touch

Feedback from our readers is always welcome. To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/businesscenter>



**Piracy:** If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at [copyright@packtpub.com](mailto:copyright@packtpub.com) with a link to the material.

**If you are interested in becoming an author:** If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit <http://authors.packtpub.com>.

## Download a free PDF copy of this book

Thanks for purchasing this book!

If you'd like to read on the go, or your eBook purchase is not compatible with your device, you can now get a DRM-free PDF version with any Packt book at no extra cost.

To do so, follow these simple steps:

1. Scan the QR code or visit the link below



<https://packt.link/free-ebook/9781804612637>

2. Submit your proof of purchase
3. That's it! We'll send your free PDF and other benefits to your email directly

## **Share your thoughts**

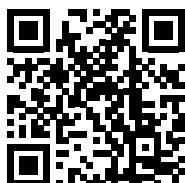
If this book is helping you improve your skills, we'd strongly suggest leaving a review on [Amazon.com](https://www.amazon.com). This helps us know if you like our work and if the chapter content has been valued, and also helps the buyers on Amazon know if the book is right for them.

So, everyone else benefits from your review - we wouldn't want you to miss out. You can now reach out to [review@packt.com](mailto:review@packt.com) with a screenshot of your review and the book URL, and we'll send you a \$5 voucher for your next Packt purchase. Thank you in advance for engaging with us, we are excited to see your review!

## **Learn more on Discord**

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/businesscenter>



---

# Section I

---

## The Basics

This section will familiarize you with the basics of Microsoft Power Platform. After completing this part, you will have a full understanding of the structure and purpose of the Microsoft Power Platform and the Microsoft Dynamics 365 ecosystem, as well as the two other Microsoft cloud offerings – Microsoft 365 and Microsoft Azure – since a typical enterprise Power Platform solution is often successfully combined with the strengths and capabilities of these other two Microsoft cloud services.

This section comprises the following chapters:

- *Chapter 1, Microsoft Power Platform and Microsoft Dynamics 365 Overview*
- *Chapter 2, Microsoft 365 and Microsoft Azure Overview*



# 1

## Microsoft Power Platform and Microsoft Dynamics 365 Overview

The **Microsoft Power Platform** is a quickly growing collection of technologies, frameworks, solutions, and products. In this chapter, you will learn about the **structure** and **modules** of the Microsoft Power Platform to make qualified decisions about the components necessary for your business solution. This will give you a good understanding of all the components of the Power Platform and will enable you to draw a high-level overview of which components you will need to fulfill your business requirements.

In this chapter, we're going to cover the following main topics:

- Introducing Contoso Inc.
- Microsoft Power Platform
- Microsoft Dynamics 365 CRM applications
- Microsoft Dynamics 365 ERP applications
- Microsoft Dynamics 365 AI, AR, and other modules
- Microsoft Power Platform licensing overview
- A practical example

First, we will introduce “Contoso Inc.,” an example company that we will revisit many times.

## Introducing Contoso Inc.

**Contoso Inc.** is a fictitious global manufacturing and retail company with headquarters in Seattle, Washington, and a number of regional subsidiaries on all continents. Contoso Inc. works in the industry of manufacturing and implementing large machines and factories as well as producing consumer electronic goods. Furthermore, Contoso Inc. operates a chain of retail stores around the globe together with an e-commerce sales channel.

This fictitious company will serve as an example Power Platform customer to present the practical implementation of the concepts presented in this book. In this chapter, Contoso Inc. will familiarize itself with the Power Platform and Dynamics 365 to decide what components will suit their business requirements.

## Introducing Microsoft Power Platform

In this section, you will learn the structure of the Microsoft Power Platform to understand the background technology on which Power Apps as well as all Microsoft Dynamics 365 applications run.

Microsoft made a big shift when it introduced the Power Platform. For seasoned Microsoft Dynamics CRM and, later, Dynamics 365 CE experts, the *Dynamics* product is no longer the centerpiece of this product line but rather a *first-party* app running on the Power Platform. The change can be illustrated in two diagrams. The first diagram shows the Microsoft Dynamics 365 high-level architecture before the Power Platform was introduced (not all applications are included for brevity):



Figure 1.1: Microsoft Dynamics 365 before the Power Platform

The second diagram presents the dramatic increase in the complexity and number of various new products and technologies that are part of the Power Platform today (not all applications are included for brevity):

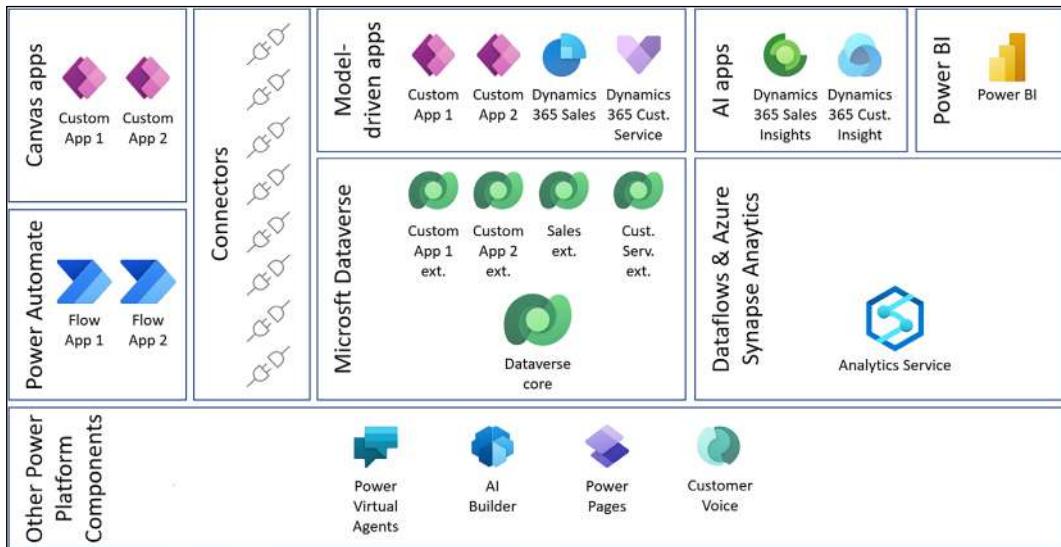


Figure 1.2: Microsoft Power Platform

As you can see in the above diagram, the introduction of the Power Platform has added a complex composition of various services and components. Let's summarize the most important:

- **Microsoft Dataaverse**
- **Power Apps**
- **Model-driven apps and canvas apps**
- **Power Automate**
- **Power Virtual Agents**
- **Power BI**
- **Power Pages**
- **AI Builder**
- **Data connectors**
- **On-Premises Data Gateway**

This new Microsoft Power Platform philosophy suggests that a potential user of a Microsoft-based business solution will need to undertake more evaluations and decision-making to find the best solution for their business requirements. Some example evaluations are as follows:

- Will my workload be better covered by some of the Microsoft Dynamics 365 applications, or should I develop my own Power Apps application?
- Do I need mobile applications at all, and if so, can I use the standard Microsoft Dynamics 365 app, or do I need to develop my own apps using the *canvas apps technology*?
- Do I need to build automation for my business application using any costly legacy integration platform or another Microsoft/third-party integration solution, or can I use *Microsoft Power Automate*?
- Can I use some of the large number of public connectors for canvas apps and Power Automate or do I need to develop my own custom connector?
- What are the most effective licensing options to cover my business requirements?

In the following sections and chapters, you will learn what Microsoft Dataverse is and what the new role of the Microsoft Dynamics 365 apps is in the Power Platform. We will also explore how you can build mobile apps for an internal audience easily and with no code, as well as look at cross-platform automation and what the licensing options are for the various Power Platform components.

## **Introducing the Common Data Model and Microsoft Dataverse**

There are two very important concepts introduced by Microsoft as part of the Power Platform, which move toward a standardized data model for business applications as well as provide a technical implementation of this model for practical use when building applications. The two concepts are the **Common Data Model (CDM)** and **Microsoft Dataverse**, and it is important to understand what these two concepts are and what the difference between them is.

### **Introducing the Common Data Model**

The **CDM** is a standardized data model consisting of a metadata system and data schemas. The CDM was developed with the goal of providing a common platform facilitating data integration and application development. It was also presented by Microsoft together with Adobe and SAP as part of the **Open Data Initiative (ODI)**. The expectation is that the ODI will welcome more partners and that all contributing parties will work on extending and further standardizing the CDM.

The CDM consists of a set of *core* tables, which are not directly related to any particular workload, together with a lot of additional tables grouped into typical workloads such as sales, service, and healthcare. An additional CDM extensibility option is the growing set of Microsoft industry accelerators. Currently, there are accelerators for banking, healthcare, education, non-profit, automotive, media, and more.

## Introducing Microsoft Dataverse

**Microsoft Dataverse** can be understood as an implementation of the CDM for the purpose of hosting data for Power Platform applications. But Microsoft Dataverse is much more than just a database. It consists of the following main components:

- Tables with the underlying structure of columns
- Relationships between tables
- User interface elements used in model-driven apps (views, forms, charts, and dashboards)
- Global (table-independent) choices, which can be repeatedly used in several tables
- Automations (business rules, business process flows, and workflows)
- Security concept elements (for example, business units, security roles, and column security profiles)
- Custom development capabilities (API, server-side, and client-side extensibility models)

Microsoft Dataverse is the foundation for building model-driven apps. The approach is to first configure the whole Microsoft Dataverse data model, create all elements, and then configure a model-driven app using the necessary subset of Microsoft Dataverse elements. Depending on the purchased licenses, Microsoft Dataverse is extended either by a Microsoft first-party application from the Microsoft Dynamics 365 family of products, any third-party partner applications, or within a user's own configuration capacity.

## Introducing model-driven apps

**Model-driven apps** are one of the two interactive end user application types that can be developed in the Power Platform. Historically, model-driven apps were the Microsoft Dynamics 365 applications themselves. The philosophy of model-driven apps has, however, changed compared to Microsoft Dynamics 365 and is based on the following capabilities:

- A model-driven app can be either a first-party application (any of the Microsoft Dynamics 365 apps), a third-party ISV application purchased from a partner or on AppSource, or a self-developed application.

- A model-driven app is based on Microsoft Dataverse.
- There can be only one Microsoft Dataverse database in a single environment but multiple model-driven apps of any of the mentioned types.
- All model-driven apps share the same data or subsets of data in Microsoft Dataverse.
- Model-driven apps run primarily on a PC in a browser, but can also be used on mobile devices within platform-specific mobile apps.



You will learn details of the Power Platform environment in *Chapter 3, Understanding the Microsoft Power Platform Architecture*.

A model-driven app is technically a very simple unit, consisting of only two components:

- A model-driven app, specified with a few parameters, for example, name and URL
- A model-driven app navigation pane called the **site map**

An example of the model-driven app user interface is shown in the following screenshot:

A screenshot of a Microsoft Power Apps model-driven app. The top navigation bar shows "Power Apps" and "Innovation Challenge". The left sidebar has sections for "Dashboards" (Home, Recent, Pinned), "Innovation" (Challenges, Ideas, Team Projects), and "ABC". The main area is titled "Active Challenges" and lists the following data:

	Name	Number...	Launch...	Accept new...	Close Date	Awards issu...	Actions
1	Connected Operations	5	3/12/2018	4/29/2018	5/5/2018	No	→
2	Enterprise sustainability	4	4/16/2018	4/30/2018	5/3/2018	No	
3	Connected products	3	10/1/2018	10/30/2018	11/4/2018	No	
4	3D Printing	2	3/1/2018	4/30/2018	5/3/2018	No	
5	Smarter manufacturing	2	5/15/2018	5/29/2018	6/2/2018	No	
6	Big data	2	12/1/2018	12/30/2018	1/4/2019	No	
7	Servitization	1	8/1/2018	8/29/2018	9/2/2018	No	
8	Renewable energy	1	9/1/2018	9/30/2018	10/4/2018	No	

Figure 1.3: Example model-driven app

Here we can see the site map navigation pane on the left side of the screen, with the app itself in the center.

All other components of a model-driven app are stored within the Microsoft Dataverse environment and should exist prior to creating a model-driven app. The end-to-end process of creating a model-driven app can be divided into the following steps:

1. Provision a Power Platform environment with the Microsoft Dataverse.
2. Create the data model (tables with columns and relationships).
3. Create the user interface (views, forms, charts, and dashboards).
4. Create the automations (business rules, business process flows, and workflows).
5. Create a new model-driven app.
6. Create a site map.
7. Select components for the model-driven app.
8. Save, validate, publish, and play.

## Introducing canvas apps

Canvas apps are a younger member of the Power Platform family and are primarily intended to be used on mobile devices rather than on PCs. Historically, canvas apps evolved from the Microsoft Siena project of 2014. The philosophy and capabilities of canvas apps are very different from model-driven apps:

- Canvas apps can be created in a Power Platform environment without Microsoft Dataverse.
- Canvas apps are designed very much like apps for mobile devices, focusing on the user interface.
- The business logic in canvas apps is implemented using an Excel-like expressions language called **Power Fx**.
- Canvas apps can be connected to various data sources using *connectors*, as described in the information box below. Right now, there are more than 600 publicly available connectors, and you can easily develop your own custom connector if no public connector is suitable for the required technology.
- Canvas apps can run on a mobile device within the Power Apps mobile app. They can run on a PC in a browser or can be embedded in websites, SharePoint sites, Power BI, Teams, or even model-driven apps.

### Introducing data connectors



**Data connectors** in Power Platform are used to support the low-code/no-code character of the Power Platform when connecting to various IT systems and solutions. Instead of developing complex interface connections, the citizen developer can just select the right connector, configure the settings, and start communicating with the systems.

An example of the canvas app user interface is shown in the following screenshot:



Figure 1.4: Example canvas app

Here we can see an interface that would be logical to use on desktop or mobile. Within the Power Apps environment, you would be able to select different areas of the screen and modify their shape, size, and color, as well as select buttons and develop expressions related to them that affect how users interact with the app.

For an accelerated adoption of canvas apps, Microsoft offers a wide variety of canvas app templates within the canvas apps designer tool.

The end-to-end process of creating a canvas app can be divided into the following steps:

1. Provision a Power Apps environment with or without Microsoft Dataverse.
2. Create a canvas app.
3. Connect to data sources using *connectors*.
4. Create the user interface (screens with controls like galleries, forms, and many other controls).
5. Create the business logic using the expression language *Power Fx*.
6. Save, validate, play, and share.

## Introducing Power Automate

**Power Automate** (previously **Flow**) is the automation engine within Microsoft Power Platform. The purpose of Power Automate is to build automation flows across a wide variety of systems and technologies in a low-code style using a very intuitive graphical user interface. The underlying technology of Power Automate is Microsoft Azure Logic Apps and the use of connectors, as well as the graphical design of flows, are very similar. A Power Automate flow generally consists of a trigger and business logic, which in turn consists of flow control elements (conditions, switches, and loops) and actions, implemented mainly using *connectors*, most likely as in canvas apps. The following types of flows are available:

- **Automated flows:** These are triggered in the background by an event trigger usually coming from a connector.
- **Button flows:** These are triggered manually by a button and can take manual data input. These flows can run on PCs and mobile devices within a specific Power Automate mobile app.
- **Scheduled flows:** These are triggered in the background by a timer or scheduler trigger.
- **Desktop flows:** These are used for recording and automating manual steps on various legacy software.

For an accelerated adoption of Power Automate, Microsoft offers a wide variety of Power Automate templates within the flow designer tool.

The end-to-end process of creating a Power Automate flow can be divided into the following steps:

1. Provision a Power Apps environment with or without Microsoft Dataverse.
2. Create a Power Automate flow.
3. Define the trigger type and trigger parameters.

4. Create the business logic using the *graphical designer*, and configure the actions appropriately.
5. Save, validate, test, and share.

## Introducing Power Virtual Agents

**Power Virtual Agents** is the latest member of the Microsoft Power Platform product family. The purpose of the Power Virtual Agents technology is to enable the creation of chatbots using a *graphical interface* and *no-code approach*, and so open the world of chatbots to everyday business users without specific programming skills. The following are the capabilities of Power Virtual Agents:

- The chatbots are developed in a graphical designer without any coding requirements.
- The designer defines conversational topics, business logic, and actions for the conversation. The actions can be implemented using Power Automate.
- The chatbot can be integrated with a variety of environments, including websites, mobile apps, Teams, Skype, and Cortana, as well as various non-Microsoft systems, such as Facebook, Slack, Telegram, and Twilio.

The end-to-end process of creating a Power Virtual Agents chatbot can be divided into the following steps:

1. Create a Power Virtual Agent bot.
2. Create topics.
3. Create the conversational logic using *questions*, *messages*, and *actions*.
4. Specify the end of the conversation by either a survey or transfer to an agent.
5. Save, validate, and test.
6. Publish the bot into a required channel.

## Introducing Power BI

**Power BI** is a collection of a data platform, cloud services, applications for PC and mobile devices, and connectors working together to provide an analytical and reporting solution. From a consumer point of view, Power BI provides Power BI apps, which consist of **reports** and **dashboards**. This content can be consumed in a browser on PCs as well as on Power BI mobile apps.

For designers and developers, Power BI offers the following capabilities:

- The Power BI cloud service, where the published content runs

- Designer and developer tools to prepare the content, such as Power BI Desktop and Power BI Report Builder
- Tools for advanced topics such as creating custom visuals or using the Power BI API

Key concepts of Power BI for Microsoft Power Platform solutions are as follows:

- Power BI reports and dashboards can use data from Microsoft Dataverse alone as well as combined with data from various other sources.
- Power BI reports and dashboards can be embedded in model-driven apps as well as canvas apps and Power Pages portals.
- Canvas apps can be embedded in Power BI reports and dashboards.

## **Introducing On-Premises Data Gateway**

**On-Premises Data Gateway** is a specific software solution enabling the use of on-premises data sources within various cloud services, such as Power Apps, Power Automate, and Power BI, as well as some Microsoft Azure services. On-Premises Data Gateway needs to be installed on a local infrastructure and configured to expose the required on-premises data sources to the cloud. The benefit of this solution is that there is no inbound connection from the cloud to the user's own data center; the connection is always established outbound.

## **Introducing AI Builder**

AI Builder is one of the latest members of the Microsoft Power Platform product family. The purpose of AI Builder's technology is to enable the creation of AI components using a *graphical interface* and a *no-code approach*, opening the world of AI to everyday business users without specific scientific and programming skills. Here are the characteristics of AI Builder:

- AI solutions are developed in a graphical designer without any AI or programming requirements.
- The user selects one of the ready-made AI models the platform offers, provides data for training the model, trains the model, and publishes the solution.
- The AI solution can be used from Power Automate flows as well as from model-driven apps or canvas apps to infuse AI-processed content into an automation or application.

AI Builder provides AI models like document processing, invoice processing, and receipt processing for the automated extraction of text blocks from documents stored as image files. Further, there are multiple models for processing text blocks, including sentiment analysis, text translation, and language detection.

## Introducing Power Pages

Power Pages recently evolved from Power Apps portals, which in turn historically evolved from *Microsoft Dynamics 365 portals*. The purpose of **Power Pages** is to provide external-facing websites connected with Microsoft Dataverse data for users outside of their own organization.

An example of the Power Pages app user interface is shown in the following screenshot:

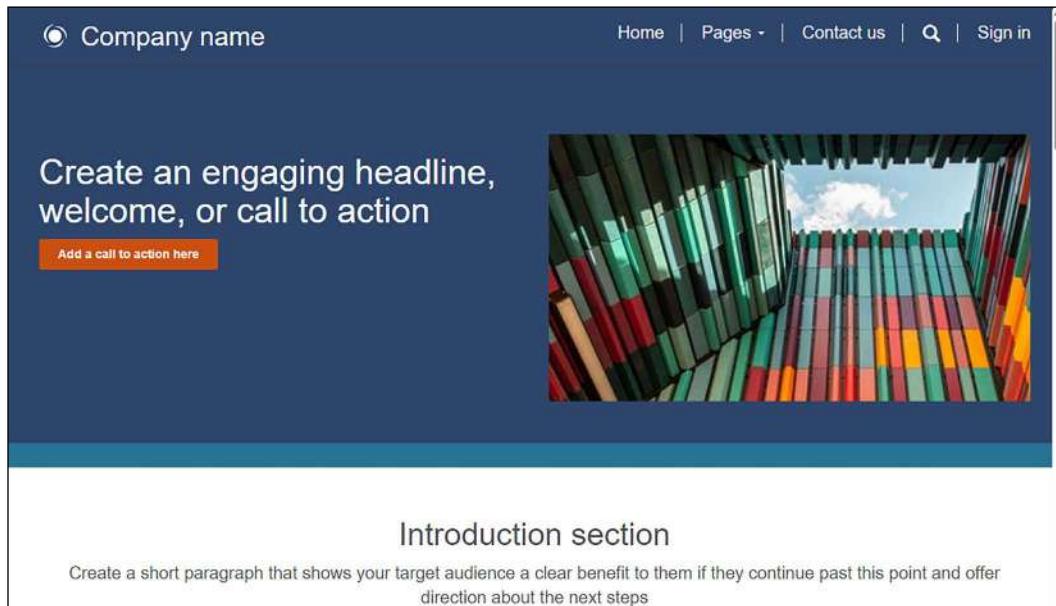


Figure 1.5: Example Power Pages app

Portals made using Power Pages are the only Microsoft Power Platform technology that are really open to public and even anonymous access. Power Pages portals have the following capabilities:

- The portals run on Microsoft Azure services, but the content of the portals is completely configured within model-driven apps.
- Power Pages portals expose selected data from Microsoft Dataverse to anonymous or registered and authenticated users.
- Power Pages portals offer a wide variety of authentication possibilities for external visitors.

## Introducing Microsoft Dynamics 365 CRM applications

Microsoft Dynamics 365 CRM technology was historically the foundation for Microsoft Dataverse, but in the current concept, Microsoft Dataverse is the foundation and the Microsoft Dynamics 365 CRM applications are called **first-party** Power Apps. Each of the applications, when provisioned, extends the Microsoft Dataverse database with the workload-specific data model and provides one or more model-driven apps for managing the respective workload. The applications cover the typical CRM workloads of sales, marketing, and customer, field, and project services. The Microsoft Dynamics 365 CRM applications are available also as an on-premises deployment, except for Marketing and Project Operations. In this section, you will learn the capabilities of all five main Dynamics 365 CRM applications.

### Microsoft Dynamics 365 Sales

The Microsoft Dynamics 365 Sales app (also called **Sales Hub**) has the following capabilities:

- Customer management
- Lead and opportunity management
- Quote and order management
- Product catalog and sales literature
- Competitor management
- Goal management
- Playbooks
- Sales Insights
- Surveys
- Reporting and analytics

### Microsoft Dynamics 365 Marketing

The Microsoft Dynamics 365 Marketing app has the following capabilities:

- Customer journeys
- Marketing content
- Segments
- Marketing emails

- Marketing forms
- Marketing pages
- Lead management
- Internet marketing
- Social media marketing
- Event management
- Surveys
- Reporting and analytics

## **Microsoft Dynamics 365 Customer Service**

The Microsoft Dynamics 365 Customer Service app (also called **Customer Service Hub**) has the following capabilities:

- Case management
- Queues
- Entitlements
- Service-level agreements
- Knowledge management
- Surveys
- Multi-session capability
- Omnichannel capability
- Reporting and analytics

## **Microsoft Dynamics 365 Field Service**

The Microsoft Dynamics 365 Field Service app has the following capabilities:

- Work order management
- Resource management
- Universal resource scheduling
- Agreements
- Inventory management
- Asset management
- Field service mobile

- Connected Field Service for IoT integration
- Surveys
- Reporting and analytics

The Dynamics 365 Connected Field Service *ad-on* extends the capabilities of the solution with IoT-enabled devices for proactive and remote maintenance.

## **Microsoft Dynamics 365 Project Operations**

The Microsoft Dynamics 365 Project Operations app has the following capabilities:

- Project-based opportunity management
- Project management
- Resource management
- Project service scheduling
- Project time and expense tracking
- Project billing
- Reporting and analytics

## **Introducing Microsoft Dynamics 365 ERP applications**

Although the Microsoft Dynamics 365 ERP applications are not in the scope of this book, it is important to have a basic understanding of the capabilities of these modules to make informed decisions when designing a future business solution. The Microsoft Dynamics 365 ERP applications are not based on the Microsoft Dataverse platform and are not considered model-driven applications. Some of the applications are also available as an on-premises deployment. In this section, you will learn the capabilities of all five main Dynamics 365 ERP applications.

## **Microsoft Dynamics 365 Finance**

The Microsoft Dynamics 365 Finance app is dedicated to midsize to large multinational customers and has the following capabilities:

- General ledger
- Accounts receivable
- Accounts payable
- Budget and forecasting

- Project accounting
- Invoicing and billing
- Fixed assets
- Cash and bank management
- Expenses
- Credits and collections
- Reporting and analytics
- Compliance management

## **Microsoft Dynamics 365 Supply Chain Management**

The **Microsoft Dynamics 365 Supply Chain Management** app is dedicated to midsize to large multinational customers and has the following capabilities:

- Product management
- Master planning
- Sales order management
- Procurement and sourcing
- Asset management
- Warehouse management
- Transportation management
- Service management
- Manufacturing
- Reporting and analytics

## **Microsoft Dynamics 365 Commerce**

The **Microsoft Dynamics 365 Commerce** app is dedicated to midsize to large multinational customers operating in the retail business and has the following capabilities:

- Unified commerce
- Modern POS
- Merchandise management
- Inventory
- Customer loyalty

- Channel management
- e-Commerce
- Reporting and analytics

## **Microsoft Dynamics 365 Human Resources**

The Microsoft Dynamics 365 Human Resources app has the following capabilities:

- Organizational management
- Employee and manager self-service
- Employee performance and development
- Goals, skills, training, and certifications
- Compensation and benefits
- Leave and absence
- Reporting and analytics

## **Microsoft Dynamics 365 Business Central**

The Microsoft Dynamics 365 Business Central app is dedicated to small to midsize local customers with no complex structures and business requirements and has the following capabilities:

- Financial management
- Supply chain management
- Sales and service management
- Project management
- Operations management
- Reporting and analytics

## **Introducing Microsoft Dynamics 365 AI, MR, and other modules**

For a long time, Microsoft has been heavily investing in bringing advanced capabilities into the business solutions product line by leveraging modern technologies such as artificial intelligence and mixed reality. The result is a large and always-growing number of Dynamics 365 addons and separate applications extending the business solution modules with analytical and insight capabilities as well as unique capabilities provided by the HoloLens hardware.

## Microsoft Dynamics 365 Customer Insights

Microsoft Dynamics 365 Customer Insights is a **customer data platform (CDP)** solution intended to bring a comprehensive 360° view of the customer by combining master data with transactional, observational, and behavioral data from various data sources using connectors. Microsoft Dynamics 365 Customer Insights has the following capabilities:

- Data source configuration
- Data unification (**mapping, matching, and merging**)
- Data enrichment
- Customer card – providing a 360° view of the customer's data with aggregated information from multiple sources
- Customer segmentation – creating segments for Microsoft Dynamics 365 Marketing
- KPI measures
- Intelligence and predictions
- Connections to Power Apps, Power Automate, and Power BI

## Microsoft Dynamics 365 Sales Insights

Microsoft Dynamics 365 Sales Insights is a set of AI-driven capabilities for Dynamics 365 Sales, from which some are included in the Sales app for free, while others are based on additional licenses:

Free with Dynamics 365 Sales	Advanced features for Dynamics 365 Sales
<ul style="list-style-type: none"><li>• Assistant</li><li>• Auto Capture</li><li>• Email Engagement</li></ul>	<ul style="list-style-type: none"><li>• Relationship Analytics</li><li>• Predictive Leads Scoring</li><li>• Predictive Opportunity Scoring</li><li>• Notes Analysis</li><li>• Who Knows Whom</li><li>• Talking Points</li></ul>

Figure 1.6: Dynamics 365 Sales Insights capabilities

The capabilities significantly improve the productivity of sales staff, providing automated guidance, notifications, reminders, conversation starters, suggestions for taking action, and advanced sales analytics dashboards and overviews.

## **Microsoft Dynamics 365 Connected Spaces**

**Microsoft Dynamics 365 Connected Spaces** is an *IoT- and AI-based* solution for analyzing and improving store operations by collecting information from video cameras and IoT-enabled devices to understand customer behavior and situations that need attention. The solution analyzes the signals and situations and creates notifications and alerts.

## **Microsoft Dynamics 365 Fraud Protection**

**Microsoft Dynamics 365 Fraud Protection** is an add-on for the Dynamics 365 Commerce solution to provide payment fraud protection and protection of other business scenarios when using e-commerce. The solution analyzes customer identities, e-commerce transactions, payments, and other activities and, with the support of AI, identifies risks and recommends actions.

## **Microsoft Dynamics 365 Remote Assist**

**Microsoft Dynamics 365 Remote Assist** is a mixed-reality solution used to improve the efficiency of on-site maintenance and repair work, where an on-site service technician equipped with the HoloLens device can benefit from the support and guidance of remote experts. The solution is used as an extension of the Microsoft Dynamics 365 Field Service app. There is also a mobile version of this solution, using a mobile app for iOS or Android instead of HoloLens.

## **Microsoft Dynamics 365 Guides**

**Microsoft Dynamics 365 Guides** is a mixed-reality solution used to improve the efficiency of learning on the job for employees new to a particular job role or providing complex procedures. The employee can use the *HoloLens device*, where holographic work instructions including text, images, videos, or 3D models are presented to them and guide their work.

## **Microsoft Dynamics 365 Product Visualize**

**Microsoft Dynamics 365 Product Visualize** is a mixed-reality solution used in the sales process for presenting, discussing, and configuring complex products in front of customers to accelerate and simplify the sales process. The solution can be used on *ARKit*-compatible mobile devices using the *iOS* operating system. The solution is directly integrated with Microsoft Dynamics 365 Sales and Microsoft SharePoint.

## Microsoft Dynamics 365 Unified Service Desk

Microsoft Dynamics 365 Unified Service Desk is a framework for building call-center applications, integrating Microsoft Dataverse applications with existing legacy and third-party applications of various types (web, desktop, Java, and mainframe), and offering frontend automation workflows, session management, and telephony integration to build an integrated agent desktop solution. The solution consists of an installable desktop application and configuration and settings capabilities within the Microsoft Dynamics 365 Customer Service app.

## Microsoft Power Platform licensing overview

The purpose of this section is to provide a brief overview of the licensing possibilities of the Microsoft Power Platform components including high-level price tags. Licensing details, which are very complex and change frequently, can be found in the various licensing guides Microsoft regularly publishes:

- **Trials:** For almost all Microsoft Power Platform components, there is a possibility to provision a free trial, usually limited to 30 days.
- **Power Apps Developer Plan:** Specific free-of-charge but permanent license for learning and individual development. Cannot be used for production purposes and is limited to a single user.
- **Power Apps/Power Automate for Office 365:** These licenses allow building apps and flows using the standard data connectors only, based on the Office 365 services.
- **Power Apps/Power Automate for Dynamics 365:** These licenses are included in most Dynamics 365 licenses and allow the creation of apps or flows using the Dynamics 365 data model only.
- **Power Apps standalone plans:** These licenses allow the creation of Power Apps using all capabilities, including the premium data connectors. The pricing point for these licenses is \$5-\$20 per user and month.
- **Power Automate standalone plans:** These licenses allow the creation and use of Power Automate flows using all capabilities, including the premium data connectors. The pricing point for these licenses starts at \$15 per user and month.
- **Power Virtual Agents:** The licenses are per tenant and are based on the number of chatbot sessions. Pricing is currently at \$200 per 2,000 chatbot sessions.
- **Power Pages portals:** The licensing is based on the number of anonymous and authenticated visits per month. Pricing is currently at \$200 per 100 login sessions and \$100 per 100,000 anonymous page views.

- **AI Builder:** The licensing is quite complex and is based on *units*, offering 1,000,000 service credits. One unit currently costs \$500. The exact pricing can be calculated with the help of a specific calculator, which can be found at the following URL: <https://powerapps.microsoft.com/en-us/ai-builder-calculator/>.
- **Power BI:** Power BI has multiple pricing tiers from Power BI Free to Power BI Premium. The pricing currently starts at \$13.70 per user and month.
- **Dynamics 365:** The pricing models for different Dynamics 365 applications are different. Most of the applications provide a per-user and per-month pricing, where the price points are currently between \$50 and \$120. Some other applications have different pricing models, for example, per-tenant and per-month.
- **Add-ons:** Besides the mentioned licensing schemes, there are many different add-ons, again with different pricing models.



For further details on the Power Platform and Dynamics 365 licensing, please refer to the licensing guides:

Power Platform: <https://learn.microsoft.com/en-us/power-platform/admin/pricing-billing-skus>

Dynamics 365: <https://dynamics.microsoft.com/en-us/pricing/>

The preceding overview can serve as a first reference point when deciding on the products to purchase.

Let's take a look at two examples where paying to make a Power Platform solution is more or less feasible compared to buying a pre-made Dynamics 365 solution.

Let's imagine you need a simple relationship management solution with some basic sales management capabilities. Is it better to purchase Dynamics 365 Sales, which starts at about \$65 per user, or is it sufficient to choose a Power Apps single-app license for \$10 per user for creating simple model-driven apps with certain simple extensions?

Alternatively, when you need complex marketing management, does it make sense to choose a Power Apps single-app license for \$10 per user and invest months of complex development to implement all of the requirements or rather purchase Dynamics 365 Marketing for \$1,500 per month, where all your requirements are covered out-of-the-box?

These situations are very different and would lead to different decisions, when considering the costs of licenses versus the labor effort to implement the solution.

For the first situation, you will most likely end up using the Power Platform approach to make a simple application, which might be more cost effective compared to the Dynamics 365 approach (buying a solution).

In the second situation, however, the decision could be exactly the opposite, since building a complex marketing solution from scratch using the Power Platform approach would certainly end up with multiple thousands of hours of implementation costs, whereas the Dynamics 365 Marketing app is readymade.

## **Contoso Inc. Power Platform commitment**

After this theoretical introduction and overview, it might be interesting to examine an example solution for our fictitious company, Contoso Inc., wanting to leverage all capabilities and maximize the benefits of using the Microsoft Power Platform components.

Contoso Inc. has analyzed all the high-level requirements for a new business solution and decided to leverage the Power Platform components to the greatest possible extent.

The various workloads Contoso Inc. is planning for implementation using Power Platform and Dynamics 365 applications are described in the following overview:

- **Dynamics 365 Sales Enterprise** and **Product Visualize** will be used by the sales department for managing B2B sales processes and providing analytical insights.
- **Dynamics 365 Marketing** will be used by the marketing department for running product campaigns and various presentations and conferences.
- **Dynamics 365 Customer Service**, **Power Pages**, and **Power Virtual Agents** will be used by the customer service department for managing the in-house and remote services.
- **Dynamics 365 Field Service**, **Connected Field Service**, and **Remote Assist** will be used by the field service department for running reactive as well as proactive field service, such as repairs and maintenance. They plan to implement the IoT extension to achieve true benefits for proactive maintenance. They also plan to implement the HoloLens-based remote service to improve the effectiveness of the field service processes.
- **Dynamics 365 Project Operations** will be used to manage the end-to-end project activities of the respective department.
- **Dynamics 365 Finance** will be used by the finance department to manage the financial accounting of every legal entity of the group.
- **Dynamics 365 Supply Chain Management** will be used by the production, procurement, logistics, and other departments to manage internal processes.

- **Dynamics 365 Commerce** together with **Connected Spaces** and **Fraud Protection** will be used to manage the activities in the retail store chain as well as to run the e-commerce business.
- **Dynamics 365 Human Resources** and **Dynamics 365 Guides** will be used by the HR department to manage the HR processes as well as the learning offerings for the employees.
- **Dynamics 365 Customer Insights** will be used to manage a 360° view of the customer including data from all available IT systems in the organization. Further, it will be used to generate specific segments for Dynamics 365 Marketing.
- **Power Apps** will be used by Power users and citizen developers in the organization to build small single-purpose applications.
- **Power Automate** will be used to build various cross-application automations.
- **Power BI** will be used to build all required analytical and reporting solutions. Contoso Inc. decided to purchase the Power BI Premium per-capacity license to be able to build an unlimited number of complex solutions.

The presented solution overview demonstrates that Contoso Inc. can cover many of its key business capabilities using Power Platform and Dynamics 365. In the subsequent chapters, Contoso Inc. will dive deeper into the requirements and refine the solution with further details.

## Summary

In this chapter, you learned the basics about the Power Platform and its components and capabilities, as well as the capabilities of the broad and ever-growing family of Dynamics 365 applications. With this knowledge, you will be able to decide on the most suitable Power Platform components for your needs (after analyzing a requirements catalog).

In the next chapter, we will present an overview of the two other Microsoft cloud products, Microsoft 365 and Microsoft Azure, in the context of Power Platform solutions.



# 2

## Microsoft 365 and Microsoft Azure Overview

Microsoft Power Platform solutions always use parts of the two other Microsoft cloud offerings: **Microsoft 365** and **Microsoft Azure**. Some of the services are used implicitly since they support the Power Platform's basic functionality, such as **Microsoft Azure Active Directory**, but many of those services are used to benefit from the simple existing integration and increased business functionality. In this chapter, we will provide an overview of all the Microsoft 365 and Microsoft Azure components relevant to a Microsoft Power Platform solution to better understand the added value of the combined power of the three Microsoft cloud services.

In this chapter, we're going to cover the following main topics:

- Introducing Microsoft 365
- Introducing Microsoft Azure
- Microsoft 365 and Microsoft Azure licensing overview

### Contoso Inc. cloud maturity

Contoso Inc., our fictitious company, has recognized that they are at the beginning of their cloud journey and after they have decided to implement a complex Power Platform solution, they must now analyze the possibilities offered by the other two Microsoft cloud offerings in order to enhance the capabilities of the Power Platform.

They need to integrate the Power Platform with other useful cloud services and provide a foundation for integrating the new Power Platform solution into their existing IT ecosystem.

Contoso Inc. has an existing Microsoft 365 subscription, which is being enrolled across the company. Currently, however, they are only using the cloud identity for their users.

Contoso Inc. does not yet have any Microsoft Azure subscription.

They need to analyze the possibilities of the various Microsoft 365 services to be used within the new Power Platform solution and also need to perform an analysis of Microsoft Azure to better understand the various services, their capabilities, and possible uses of the Power Platform.

## **Introducing Microsoft 365**

Microsoft 365 is a commercial product name created primarily to simplify licensing, but it consists of a rather heterogeneous group of technical, cloud, and on-premises products. In this section, you will learn about the structure of Microsoft 365 and focus on the components relevant to a Microsoft Power Platform solution.

Microsoft 365 consists of three main groups of products:

- **Microsoft Windows**
- **Microsoft Office 365**
- **Microsoft Enterprise Mobility + Security**

A basic structure of the complex Microsoft 365 product offering is documented in the following diagram:

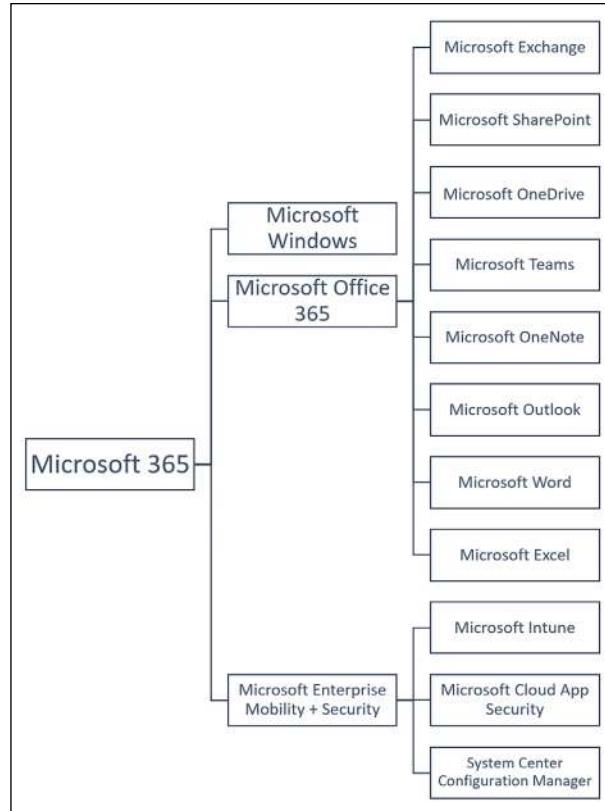


Figure 2.1: Structure of the Microsoft 365 cloud service

You'll already be familiar with the purpose of the Windows operating system, so in the next sections, we will focus on the relevant components of Microsoft Office 365 and Microsoft Enterprise Mobility + Security.

## Introducing Microsoft Office 365

**Microsoft Office 365** is a group of cloud services as well as desktop products from the Microsoft Office family. Many of the Microsoft Office 365 products are used very frequently as part of Power Platform solutions.

## **Microsoft Exchange**

**Microsoft Exchange** is a cloud email solution based on the established on-premises product. Microsoft Exchange is one of the most integrated Microsoft cloud products in Power Platform solutions, since it extends the solution with the email channel, and supports the calendar, appointments, tasks, and contact synchronization.

## **Microsoft SharePoint**

**Microsoft SharePoint** is a document management solution, collaboration space, intranet solution, and more, and is also heavily used in Power Platform solutions specifically for the integrated document management capability. The integration with SharePoint is also required for several other Power Platform integrations, for example, **OneDrive**, **OneNote**, and **Microsoft Teams**.

## **Microsoft OneDrive**

**Microsoft OneDrive** is used for private document storage as opposed to Microsoft SharePoint, where document sharing and collaboration are key capabilities. Power Platform solutions integrate with OneDrive to provide integrated document management capability for both scenarios: collaborative document management with SharePoint and private document management with OneDrive.

## **Microsoft OneNote**

**Microsoft OneNote** is a note-taking application where the user can take notes and organize them into notebooks, sections, and pages. The notes can be of various types: text, drawings, recorded audio, video, and attached files. Power Platform solutions integrate with OneNote to provide a context-based note-taking capability.

## **Microsoft Teams**

**Microsoft Teams** is a collaboration platform unifying and extending the backend services of document management, mailbox and calendar, collaboration groups, chat, telephony, and conferencing into an end user collaboration workspace. Power Platform solutions integrate with Teams to bring the Power Platform solution's structured business content into the overall collaboration capabilities of Teams.

## **Microsoft Outlook**

**Microsoft Outlook** is an email client providing capabilities for emails, calendar appointments, tasks, and contacts. Microsoft Outlook is available as a desktop application, as an app for mobile devices, and as a web client.

Power Platform solutions integrate with Outlook to provide users with a well-known product that is extended and integrated with Power Platform capabilities as a unified workspace.

## **Microsoft Word**

**Microsoft Word** is a text processing solution available as a desktop application, as an app for mobile devices, and as a cloud solution – **Word Online**. Power Platform solutions integrate with Word to provide template-based document generation enriched with Power Platform data.

## **Microsoft Excel**

**Microsoft Excel** is a spreadsheet solution available as a desktop application, as an app for mobile devices, and as a cloud solution – **Excel Online**. Power Platform solutions integrate with Excel to provide template-based document generation enriched with Power Platform data as well as data export and data import capabilities.

## **Overviewing Microsoft Enterprise Mobility + Security**

**Microsoft Enterprise Mobility + Security (EM+S)** is a group of cloud and on-premises products for security and mobility management. The group consists of various products, not all of which are relevant to the Power Platform.

### **Microsoft Intune**

Microsoft Intune is a cloud solution dedicated to **mobile device management (MDM)** and **mobile application management (MAM)**. This solution is used to enroll mobile devices, enforce device protection, data protection and compliance policies, separate private and corporate data for private devices used for business purposes, and so on. Microsoft Intune is used to manage mobile devices with installed Power Apps, Power Automate, Power BI, and Dynamics 365 mobile apps to manage encryption and app-to-app communication. It can also be used to uninstall and wipe out Power Platform business apps for lost and stolen devices or in case an employee leaves the company.

### **Microsoft Cloud App Security**

**Microsoft Cloud App Security** is a cloud solution dedicated to identifying risks and cyber threats and protecting customers' cloud service environments. The solution has connectors to many Microsoft and third-party cloud services, including Microsoft Dynamics 365. In order to include Dynamics 365 in the Microsoft Cloud App Security protection, Dynamics 365 auditing must be enabled.

## System Center Configuration Manager

**System Center Configuration Manager** (now part of the *Microsoft Endpoint Manager suite*) is an on-premises solution dedicated to device and application management of Microsoft Windows devices. The solution is used to enroll Windows-based server and desktop devices, install and upgrade Windows applications, enforce policies, grant access to corporate resources, and so on. **System Center Configuration Manager** can be used to manage Power Platform components that are physically installed on users' PCs, such as Unified Service Desk, or other local installations.

## Contoso Inc. using Microsoft 365

Following a detailed analysis of the current deployment and use of the Microsoft 365 subscription and the opportunities to integrate many of the Microsoft 365 services with the Power Platform, Contoso Inc. has made a decision regarding a number of usage scenarios.

- **Microsoft Exchange:** Contoso Inc. plans to establish server-side integration between Dynamics 365 applications and Exchange so as to provide an integrated email and calendar experience.
- **Microsoft SharePoint:** Contoso Inc. will establish integration between the Dynamics 365 applications and SharePoint for integrated document management, specifically in the sales department.
- **Microsoft OneDrive:** Contoso Inc. will establish integration between the Dynamics 365 applications and OneDrive to enable a private document store for confidential documents where sharing is not desired.
- **Microsoft OneNote:** Contoso Inc. will establish integration between the Dynamics 365 applications and OneNote to benefit from the capabilities of OneNote in terms of storing not just text, but also handwriting, pictures, video, and audio to bring this data into the context of Dynamics 365 records.
- **Microsoft Teams:** Contoso Inc. just deployed Microsoft Teams across the entire organization and will establish integration with Dynamics 365 to provide a unified collaboration workplace extended with Dynamics 365 data.
- **Microsoft Outlook, Word, and Excel:** Contoso Inc. plans to benefit from the integrated Dynamics 365 App for Outlook as well as from standard template capabilities with Word and Excel.
- **Microsoft Intune:** Contoso Inc. will use *Intune* to deploy and manage the Power Platform mobile apps on employees' mobile devices.

- **Microsoft Cloud App Security:** Contoso Inc. will use *Microsoft Cloud App Security* to enhance the security of the Power Platform solution by integrating this component into the solutions.

## Introducing Microsoft Azure

After describing the Microsoft 365 offering, we will now turn our attention to Microsoft Azure as the other cloud service heavily used in conjunction with Power Platform solutions.

**Microsoft Azure** is a Microsoft cloud platform consisting of hundreds of different cloud services, primarily **Infrastructure as a Service (IaaS)** and **Platform as a Service (PaaS)** types. There are around 60 Azure regions worldwide covering around 140 countries, with new data centers appearing every year.

In this section, you will get an overview of the Microsoft Azure services that are most relevant for a Power Platform solution. By implementing, integrating, and using these services, the Power Platform solution can be easily extended with many new capabilities, including artificial intelligence, IoT, integration, and automation.

The overview of the Azure services covered in this chapter is documented in the following diagram:

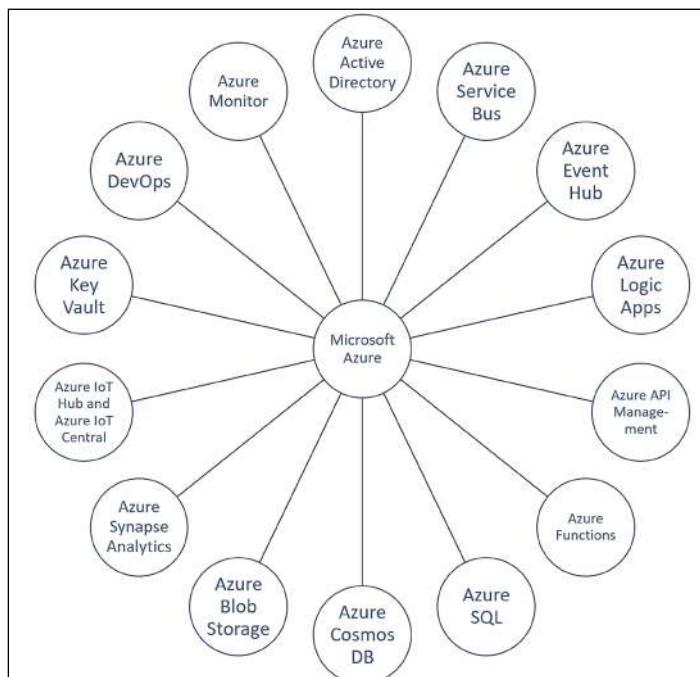


Figure 2.2: Microsoft Azure components most relevant to Power Platform

There are many other Microsoft Azure services that can be used within a Power Platform solution. However, this would go beyond the scope of this section. For now, we just need to overview the main services that are available, and cover the situations where we should use these services in a solution.

## Introducing Azure Active Directory

Azure Active Directory (AAD) is a key component of all Microsoft cloud services. In contrast to some other cloud providers, Microsoft requires every user to have an account in AAD in order to be able to use any cloud service. AAD is also a baseline for cloud service licensing since all user licenses are assigned on the AAD level. For Microsoft Power Platform, AAD is used indirectly through the Microsoft 365 administration site, which contains user management, group management, and license management as a wrapper over AAD.

There is also another cloud directory component called **Azure Active Directory B2C (Business to Consumer)**, used to manage external identities such as customers, consumers, and citizens. AAD B2C has special relevance for Power Pages, where it can be used as one of the options for authentication of external portal visitors.

In the next chapters, you will learn about AAD user provisioning, authentication types, license management automation, integration, and so on.

## Introducing Azure Service Bus

Azure Service Bus is a message broker solution, used primarily for message-based integration scenarios. Microsoft Dataverse contains a built-in integration to Azure Service Bus. This integration can be used to implement hybrid integration scenarios when there is a need to provide a secure communication channel between Microsoft Dataverse and some on-premises IT systems. We will use Azure Service bus for Dataverse integration in *Chapter 8, Microsoft Power Platform Extensibility*.

## Introducing Azure Event Hubs

Azure Event Hubs is an event ingestion and streaming solution, used primarily for large-scale event processing from several event sources and distributing the events to several consumers. Power Platform contains a built-in integration with Azure Event Hubs to support integration solutions in the same way as for Azure Service Bus.

## Introducing Azure Logic Apps

**Azure Logic Apps** is a cloud integration solution for building integration and automation solutions using the business logic capabilities and the capabilities of a large number of different connectors to many Microsoft as well as non-Microsoft products and technologies. Azure Logic Apps is the foundation for Power Automate, but the usage scenarios are slightly different, as you will learn in later parts of this book. Power Platform solutions can benefit from using Azure Logic Apps for building cloud as well as hybrid integration scenarios.

## Introducing Azure API Management

**Azure API Management** is a component for building and publishing APIs to existing on-premises or cloud solutions and offering these APIs to internal or external third-party applications. Azure API Management provides advanced capabilities for **security, authentication, quota limits, caching, and monitoring**. Power Platform solutions can use Azure API Management for various integration scenarios alone or in combination with other Azure components for protecting the Power Platform API, exposing and securing part of it to external systems, and so on.

## Introducing Azure Functions

**Azure Functions** is a platform for building and operating serverless computing applications. An Azure Functions solution can be developed in a variety of programming languages and consists of a trigger and business logic. Power Platform solutions can benefit from using Azure Functions in several ways:

- Azure Functions solutions can be triggered from a Power Platform solution using the *HTTP trigger*
- Azure Functions solutions can be triggered from an *integrated Azure Service Bus* or *Azure Event Hub*
- Scheduled jobs for a Power Platform solution can be developed using the *Timer trigger*

## Introducing Azure SQL

**Azure SQL** is basically a Microsoft SQL database engine deployed in the cloud as a PaaS solution. The user of Azure SQL does not need to maintain any software installations, just create, maintain, and use their databases. Power Platform solutions can benefit from using Azure Functions in several ways:

- Database replication of the Dataverse database into its own Azure SQL instance to get full access to Power Platform data on the database level

- Database consolidation to consolidate data from Dataverse together with various other data sources for the purpose of reporting and analytics with Power BI or other reporting tools
- Database staging for various integration scenarios involving Power Platform solutions

## Introducing Azure Cosmos DB

**Azure Cosmos DB** is a globally distributed database that allows scalability and replication across any number of Azure data centers and so provides very low latency and high response times anywhere in the world. Azure Cosmos DB is multi-model, supporting the storage of documents, key-value pairs, object graphs, and relational data. Azure Cosmos DB offers a wide variety of NoSQL APIs. Power Platform solutions can benefit from using Azure Cosmos DB in several ways:

- Azure Cosmos DB can be used as a connected database for implementing virtual tables using the free Microsoft *Azure Cosmos DB for Document DB API Data Provider* AppSource module.
- Azure Cosmos DB can be used as a global consolidation database for multiple Power Platform implementations.
- Azure Cosmos DB can be used as an archival database for a Power Platform solution using the free Microsoft *Dynamics 365 Data Archival and Retention* AppSource module.

## Introducing Azure Blob Storage

**Azure Blob Storage** is one of the storage types that can be created and used in an Azure Storage account. Azure Blob Storage is best used for massive amounts of data such as text, binary, and similar data types. Power Platform solutions can benefit from integrating with Azure Blob Storage to offload large unstructured data elements, such as record file attachments from the Dataverse database to a much less expensive storage type.

## Introducing Azure Synapse Analytics

**Azure Synapse Analytics** is a cloud enterprise analytics service, combining the power of multiple other services like Azure Data Lake, Azure SQL, Apache Spark, Data Explorer, Pipelines, and others to bring data warehouse, data integration, and big data analytics as a single integrated service. The Power Platform has a built-in integration with the Azure Synapse Analytics service to provide near real-time analytics, BI, and ML scenarios based on data coming from Microsoft Dataverse.

## Introducing Azure IoT Hub and Azure IoT Central

Azure IoT Hub and Azure IoT Central are both IoT cloud integration platforms for IoT message processing, bi-directional communication, security, analytics, routing, and monitoring. The difference between them is that Azure IoT Central is a **Software as a Service (SaaS)** solution, based on top of Azure IoT, which is a PaaS solution. Power Platform solutions can benefit from both Azure IoT Hub and Azure IoT Central when integrating an IoT ecosystem with a business solution. A typical out-of-the-box implementation is the Dynamics 365 Connected Field Service module, which can integrate IoT with Azure IoT Hub or Azure IoT Central to provide a proactive service management capability.

## Introducing Azure Key Vault

Azure Key Vault is a cloud solution for securely storing various credentials, keys, passwords, certificates, connection strings, and other artifacts that require the highest level of protection. There are two versions of Azure Key Vault: a software version and a hardware version using specific *FIPS-compliant hardware modules* for increased security. Power Platform solutions can benefit from using Azure Key Vault by storing all confidential artifacts used in various cloud solutions that are integrated with Power Platform; for example, integration solutions, APIs, and batch jobs.

## Introducing Azure DevOps

Azure DevOps is a cloud development and operations tool enabling agility, continuous planning, development, integration and delivery, source control, and monitoring. Azure DevOps is used by large development teams to develop complex software solutions but can be used at any scale. Power Platform implementation teams can benefit from Azure DevOps specifically when using the *Power Platform Build Tools* to achieve fully automated solution deployments.

## Introducing Azure Monitor

Azure Monitor is a collection of telemetry and monitoring tools to analyze the cloud as well as on-premises application performance, identify issues, and provide notifications and analytics. Power Platform solutions can include Azure Monitor capabilities, specifically Azure Application Insights and Azure Log Analytics, to monitor performance and help keep the solutions in good shape.

## Contoso Inc. using Microsoft Azure

Contoso Inc. performed a detailed analysis of Microsoft Azure and recognized the relevance of this cloud service for the planned Power Platform solution implementation. They decided to purchase an appropriate Microsoft Azure subscription and made a preliminary decision regarding several usage scenarios.

- **Authentication and single sign-on:** Contoso Inc. has recognized that using pure cloud identity as it is now with their Microsoft 365 subscription is not an enterprise-ready solution. They decided, for the purpose of full governance and true single sign-on capability, that they would integrate their highly secured on-premises active directory forest with the Azure Active Directory by deploying the Azure Active Directory Federation using *Azure Active Directory Connect*. With this integration, they will continue to leverage the investments made into securing and hardening their own infrastructure.

To authenticate their external portal users, they will implement *Azure Active Directory B2C*. This approach will be used specifically when using the Power Pages technology to build portal solutions for their customer and partners.

- **Power Platform integration:** Contoso Inc. has analyzed the existing IT landscape and identified multiple existing special purpose on-premises, as well as cloud-based IT systems, that need to be integrated with Power Platform. For this purpose, Contoso Inc. has selected the following Microsoft Azure services as possible integration technology components:
  - Azure Service Bus
  - Azure Event Hubs
  - Azure Logic Apps
  - Azure API Management
  - Azure SQL
  - Azure Key Vault
- **IoT integration:** Since Contoso Inc. plans to implement Microsoft Dynamics 365 Connected Field Service, they have decided to use *Azure IoT Hub* to implement sensor integration.
- **Monitoring:** Contoso Inc. plans to establish *Azure Monitor* as the general monitoring platform for all Power Platform solutions that are planned to be developed.
- **Application life cycle management:** Contoso Inc. plans to establish *Azure DevOps* as the general development platform for all Power Platform-based solutions.

## Microsoft 365 and Microsoft Azure licensing overview

Now that you have learned about Microsoft 365 and Microsoft Azure cloud services and their practical use for Power Platform solutions, it is important to understand the basics of the licensing of these services.

The purpose of this section is to provide a brief overview of the licensing options of Microsoft 365 and Microsoft Azure, including high-level price tags. Licensing details, which are very complex and change frequently, can be found in the various licensing guides published by Microsoft on a regular basis.

### Microsoft 365 licensing

Microsoft 365 is a collection of mainly SaaS products, which is why the licensing is user/subscription based. There are several plan-based, as well as individual product-based, licensing options:

- **Microsoft 365 Enterprise (F3, E1, E3, E5):** These plans are dedicated to enterprise customers and can be purchased directly from Microsoft or from Microsoft's partners. Prices vary between \$10–\$60 per user per month.
- **Microsoft 365 Education (A3 and A5):** These plans are dedicated to academic/school customers. Prices are approximately 10%–30% of those of the enterprise licenses.
- **Microsoft 365 Business (Business Basic, Business Standard, Business Premium, and Apps for Business):** This plan is dedicated to small and medium business customers and can be purchased only from Microsoft's partners or directly on the web. Prices are approximately \$20/user/month, and for non-profit organizations, approximately \$5/user/month.

In addition to these plans, there are a variety of options available to purchase Windows 10 or 11, as well as Office 365 and EM+S subscriptions of different types.

### Microsoft Azure licensing

Microsoft Azure is a collection of IaaS and PaaS products, and licensing and pricing are based on consumption. Azure can be purchased directly from Microsoft as well as from Microsoft Partners. It is recommended to use the *Azure Pricing Calculator* for the exact cost calculation for the various Azure services since the calculation can be quite complex. Many Azure services are priced based on several metrics, such as compute time, storage space, and network traffic. There are pricing differences depending on the particular Azure region. Significant price savings of up to 80% can be achieved by using the reserved resources, where the customer commits to using certain Azure resources for a period of 1 year or 3 years.

## **Summary**

In this chapter, you have learned the basics of the two other Microsoft cloud offerings: Microsoft 365 and Microsoft Azure, which are used heavily in Power Platform solutions. With this knowledge, you should be able to assess which Microsoft 365 and Microsoft Azure components will need to be part of your solution. When you start architecting your Power Platform solution you will need to consider their feasibility for security, artificial intelligence, IoT, integration into your existing IT landscape, data migration, and many more applications besides.

Now that we have considered the components we will need to utilize in our solution, the next step is to dive deeper into the Power Platform architecture, which will be the foundational knowledge for building any Power Platform solution. Specifically, we will learn about the Power Platform environments and structure, clients, administration, and monitoring.



## **Share your thoughts**

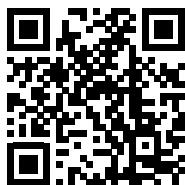
If this book is helping you improve your skills, we'd strongly suggest leaving a review on [Amazon.com](https://www.amazon.com). This helps us know if you like our work and if the chapter content has been valued, and also helps the buyers on Amazon know if the book is right for them.

So, everyone else benefits from your review - we wouldn't want you to miss out. You can now reach out to [review@packt.com](mailto:review@packt.com) with a screenshot of your review and the book URL, and we'll send you a \$5 voucher for your next Packt purchase. Thank you in advance for engaging with us, we are excited to see your review!

## **Learn more on Discord**

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/businesscenter>



---

# Section II

---

## The Architecture

In this section, we will deep dive into the Microsoft Power Platform architecture.

After completing this part, you will have a full understanding of the Power Platform architecture, tools, and techniques used for administration, configuration, customization, and application lifecycle management. After completing this part, you will be able to design and build a Power Platform solution for your own requirements.

This section comprises the following chapters:

- *Chapter 3, Understanding the Microsoft Power Platform Architecture*
- *Chapter 4, Customization and Development Tools and Techniques*
- *Chapter 5, Application Lifecycle Management*



# 3

## Understanding the Microsoft Power Platform Architecture

In this chapter, we will dive deeper into the **Power Platform architecture**. Understanding this, as well as the required technology, environments, clients, and administration and monitoring, is key to enabling an architect and Power Platform implementation team to set up the foundation of a Power Platform solution. Following the recommendations and best practices at hand can lead to robust, reliable, and highly performant solution designs and the best end user experience possible.

In this chapter, we are going to cover the following main topics:

- Understanding the Power Platform architecture
- Understanding the Power Platform clients
- Learning about Power Platform administration and monitoring
- Presenting architectural best practices
- Contoso Inc. Power Platform architecture

By the end of this chapter, you will have learned many details about the Power Platform architecture, as well as environments, technology, clients, and administration and monitoring. This will provide you with the skills you need to design a high-level architecture for your own Power Platform-based solution.

## Contoso Inc. starts architecting their planned Power Platform solution

After a detailed analysis of the available products and services in the Microsoft cloud services, that is, **Power Platform**, **Microsoft 365**, and **Microsoft Azure**, Contoso Inc. has made a series of high-level decisions with regard to which products will be part of their future solution. Now it's time to start architecting their planned Power Platform solution. Contoso Inc. will make themselves familiar with the main components of the Power Platform architecture in order to carry out the proper preparation steps for the Power Platform implementation project.

Let's start by familiarizing ourselves with the main features of the Power Platform architecture in order to be able to design our own solution architecture.

## Understanding the Power Platform architecture

**Power Platform** is one of the three Microsoft cloud services and is part of the overall Microsoft cloud infrastructure. In this section, we will focus on explaining the Microsoft cloud infrastructure, the structure of a customer cloud ecosystem, and the Power Platform technology and environments, as well as their main components. This is key to understanding the architectural principles of the Power Platform and making proper decisions about the overall architecture of your solution. First, we will describe the Microsoft cloud infrastructure to better understand the key concepts.

## Learning about the Microsoft cloud infrastructure

The **Microsoft cloud** products and solutions are operated out of Microsoft data centers, which are grouped into regions and geographies. Each region contains two or more data centers for failover safety and high availability. Microsoft has many cloud regions worldwide. For Microsoft Azure, Microsoft Office 365, and Power Platform, these regions are structured differently. For example, for Microsoft Azure in Europe, there are many separate regions, such as North Europe, West Europe, UK West, UK South, France Central, France South, Germany West Central, Germany North, Germany Northeast, Germany Central, Norway West, Norway East, Switzerland North, and Switzerland West. These can all be selected when you're provisioning Azure services.

For Power Platform cloud services, the customer cannot select the region in such a detailed granularity as for Azure. This is because – for example – the customer can select the Power Platform region *Europe* but can't decide whether it will be *North Europe* or *West Europe* like they can for Azure. Currently, the following regions are available worldwide when a customer wants to create a new Power Platform environment:

- Asia
- Australia
- Canada
- Europe
- France
- Germany
- India
- Japan
- Korea
- Norway
- South America
- Switzerland
- United Arab Emirates
- United Kingdom
- United States
- US Government (GCC)
- Preview (United States)

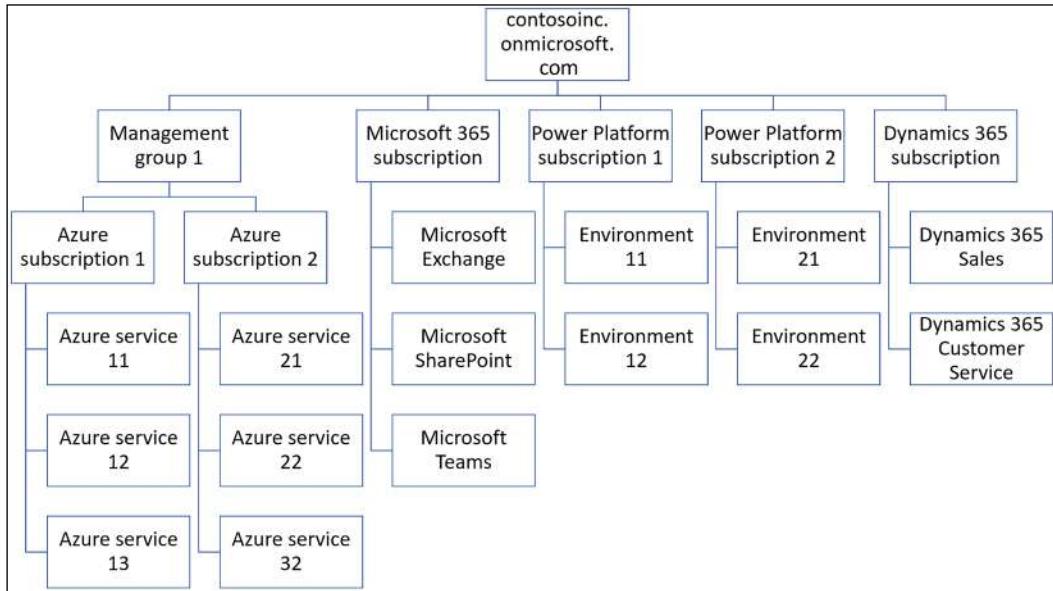
Within these regions, there is one specific region called *Preview (United States)*, which contains preview features of the Power Platform components not yet available in the other regions. Selecting this region is useful for testing the latest product features and giving feedback to Microsoft. Currently, no **Microsoft Dataverse** database can be created in this region, so no Dataverse and model-driven app preview features are available when using this environment.

## Understanding the customer cloud structure

When a customer first purchases any Microsoft cloud service, they always get a certain basic structure that is necessary for operating the service, or any other additional cloud service from Microsoft. This structure is built around a **tenant**. This tenant is technically an **Azure Active Directory** domain that's used to support the central services for each cloud solution, regardless of whether it is **Microsoft Azure**, **Microsoft 365**, or **Microsoft Power Platform**.

Within one tenant, there can be several Microsoft cloud service subscriptions. Depending on the subscription type, there can be various cloud services belonging to those subscriptions.

Let's assume our fictitious company, Contoso Inc., already uses various Microsoft cloud services. Their cloud structure could look as follows:



*Figure 3.1: Contoso Inc. structure of the Microsoft cloud services*

As we can see, a large organization can have a very complex structure consisting of many parallel cloud subscriptions from different cloud services.

The tenant itself is always located in a certain Microsoft cloud region, but any Power Platform environments are independent of the tenant's region and can be created in any Power Platform-enabled region. This can be selected by the creator.

For the purpose of Power Platform, there are certain essential central tenant services as follows:

- **User management**
- **License management**
- **Group management**
- **App registrations**
- **Office 365 Activity Logging**

In the following sections, you will learn more about these services.

## User management

Any user of any Microsoft cloud service, including Power Platform, must first be registered in the Azure Active Directory of the customer's tenant. User management in the tenant can be done in a variety of ways:

- Manual user management using the *Microsoft 365 admin center* or in the *Microsoft Azure portal*
- Synchronizing user identities automatically from the customer's on-premises Active Directory
- Using *scripting automations* with PowerShell
- Using a *third-party identity management solution*, integrated with Azure Active Directory using the Graph API

## License management

After a user has been created in Azure Active Directory, access to any cloud service is granted by assigning the respective product license. After the license has been assigned, a background process starts provisioning access to the service for the user. For license management, the same management options that are available for user management are provided.

## Group management

Groups are used for the following purposes for Power Platform solutions:

- Managing user provisioning into distinct Power Platform environments
- Managing authorization within Microsoft Dataverse applications
- Managing authorization for Power Apps canvas apps and Power Automate cloud flows

For group management, the same options for manual, scripting-based, or automated management for user or license management are available.

## App registration

**App registration** is a security feature necessary for implementing **OAuth** authentication scenarios for external applications or integrations connected with the Power Platform API.

## Office 365 Activity Logging

**Office 365 Activity Logging** is an auditing capability for Office 365. This also includes Microsoft Dataverse auditing.



#### Important note

More details about user, license, and group management, as well as app registrations, will be discussed in later chapters.

## Learning about Power Platform technology

Power Platform is a cloud service and therefore there are not many publicly available technical details about the background technology. Power Platform is operated on so-called **scale groups**, which are unified blocks of cloud infrastructure that consist of various infrastructure components necessary for running the services. There are database servers, reporting servers, web servers, app servers, integration servers, and many more, but the Power Platform customer doesn't have access to these infrastructure components. Scale groups are only located in the Power Platform-enabled cloud regions.

Due to the nature of the **Software as a Service (SaaS)** cloud model, scale groups are shared among multiple customers. One single scale group can host Power Platform environments for dozens or even hundreds of customers.



#### Important note

Large Power Platform customers can arrange with Microsoft a dedicated use of a scale group just for their own purposes without having the environments of other customers.

This approach requires certain expected and also enforced behavior standards, specifically to refrain from any activities with potential heavy performance impacts on the underlying infrastructure. Certain enforced restrictions will be described later in this chapter. It is a matter of fact that the platform usually allocates more resources for production than non-production environments.

## Understanding Power Platform environments

Power Platform solutions (except for **Power BI**) always run within Power Platform environments. An environment is a logical unit and is the foundation for creating Power Platform solutions. It also acts as a container for all the resources that are used in the solutions. In this section, you will learn about the environments and their main components.

Let's start by describing the different types of Power Platform environments before we describe the main components within them.

There are seven types of environments:

- **Default:** This is automatically created in every Power Platform licensed tenant. It can be used for evaluating, proof of concepts, and so on, but should not be used for complex solution development or production, due to lack of access control.
- **Trial:** This is a temporary environment, best suited for testing specific product features, third-party solutions, demonstration purposes, and so on.
- **Developer:** This is a specific environment, provisioned with the Power Apps Developer Plan license. This environment can have only the owner as a single user. The provisioning of this environment type needs to be performed using a specific URL: <https://apps.powerapps.com/community/signup>.
- **Sandbox:** This can be used for pre-production purposes such as development, testing, training, support, and so on. However, it is not intended for production purposes.
- **Production:** This is typically used for running a deployed solution in production.
- **Microsoft Teams:** This is a new type of environment, which is used for creating and running Power Platform solutions directly within Microsoft Teams. This environment type is created from Microsoft Teams directly in the background when a Teams user starts building a Power App or Power Automate flow.
- **Support:** This is a specific environment that cannot be created by the customer, only by Microsoft Support personnel, in order to resolve service case issues. It is usually created as a copy of the existing troublesome environment and deleted after the issue is resolved.



**Tip**

Do not consider using the developer environment type for team development in large projects. It is only suitable for individual developers.

When created in a standard way, for example in the Power Platform admin center, every environment has some basic parameters that must be specified upon creation:

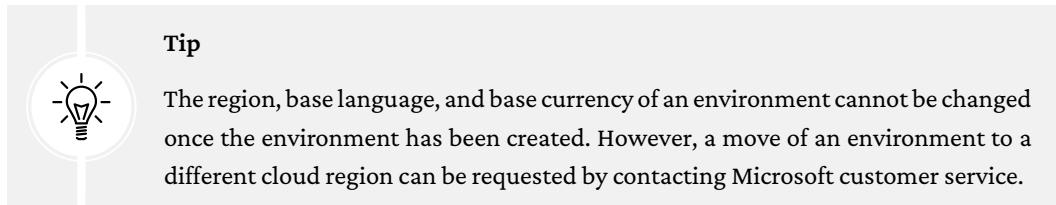
- **Display name** of the environment.
- **Environment type**, which can be either a **Sandbox**, **Production**, or **Trial** environment. The other environment types mentioned earlier are created in different ways.
- **Region**, which is selected from a list of Power Platform-enabled regions.
- **Purpose** of the environment as a free text description.

In this step, you can also decide whether a **Microsoft Dataverse database** should be created for the new environment.

If a Microsoft Dataverse database should be created, then the following additional parameters need to be specified:

- **Language:** This is the base user interface language of the Dataverse database and is used in all model-driven applications created in this environment.
- **Currency:** This is the base currency of the Dataverse database used by currency data type fields.
- You can also enable the deployment of some Dynamics 365 applications or a few sample model-driven applications with sample data. The sample apps with sample data cannot be created later, only upon the creation of the environment.
- **Security group:** At this point a connection with a security group can be established for better control of the access rights to the environment.

If a Microsoft Dataverse database isn't created for an environment, then no model-driven apps can be developed in that environment. However, it is possible to create a Microsoft Dataverse database later, for already-created environments.



### Tip

The region, base language, and base currency of an environment cannot be changed once the environment has been created. However, a move of an environment to a different cloud region can be requested by contacting Microsoft customer service.

A Power Platform environment consists of the following main components:

- Optionally **Microsoft Dataverse**
- **All artifacts created by the makers**, like apps, flows, chatbots, AI models, etc. These components will be discussed later in this book.
- **Power Platform data connectors**
- **Data Loss Prevention policies**
- **On-Premises Data Gateway** configurations

In the following sections, you will learn more about these main components of an environment.

## Microsoft Dataverse

**Microsoft Dataverse** is used to store the metadata, business data, and application artifacts of model-driven apps, like the forms, views, charts, and dashboards. The Microsoft Dataverse storage is subdivided into the following three storage types:

- **File:** Used for storing files associated with business data such as attachments of email activities, files attached to any record within an annotation, and image and file data types
- **Log:** Used for storing logging information such as auditing data or tracing data from plug-ins
- **Database:** Used for storing all other relational business data and metadata

Every customer will receive a limited tenant-wide storage capacity for the three storage types, as explained later in this chapter.

## Microsoft Dataverse for Teams

**Microsoft Dataverse for Teams** is a specific version of Dataverse, used within the Microsoft Teams environment type, to support data management in apps and flows. The main capabilities are the same as for the standard Dataverse; there are, however, various differences and limitations, specifically:

- No advanced data types like “Currency” or “Customer”
- No advanced Dataverse search
- No offline capability
- No programmability (no API, no plug-ins, and no PCF components)
- Limited size (1 million rows or 2 GB total size)
- Limited security (no customizable security roles, no business units, no column-level security, and no hierarchical security)
- No integration apart from using data connectors from apps and flows

Those limitations represent the typical usage scenarios for apps and flows running within Microsoft Teams. If the full Microsoft Dataverse capabilities are required, an upgrade is possible.

## Capacity restrictions

Within a Power Platform environment, as well as generally for the whole tenant, there are certain capacity restrictions that need to be considered when designing Power Platform solutions:

- **Storage capacity limits**

- Request limits and allocations
- API limits

These restrictions will be described in more detail in the following sections.

## Storage capacity limits

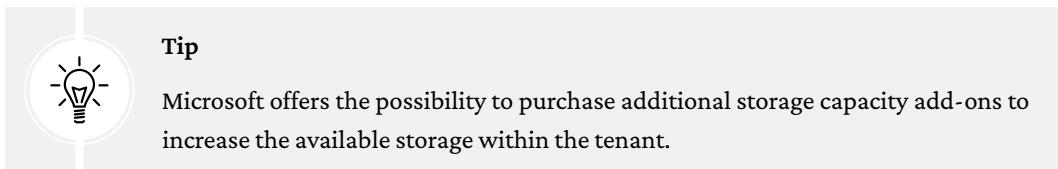
Storage capacity limits apply to all Power Platform environments created in one tenant together. There are separate capacity limits for the file, log, and database storage types, and the capacity depends on the various Power Platform license types and the number of user licenses a customer has purchased. Currently, the basic storage capacity every new customer gets with their first subscription is as follows:

- **Database:** 10 GB
- **Log:** 2 GB
- **File:** 20 GB

The following environments are excluded from the storage capacity limits:

- Trial
- Preview
- Support
- Developer

It is important to plan accordingly with regard to how many and what type of environments need to be created in order to stay within these limits.



The concept of storage capacity limits is illustrated in the following diagram:

Customer tenant	 Database storage	+	 Database storage	+	 Database storage	+	 Database storage	<= Database capacity limit
	 File storage	+	 File storage	+	 File storage	+	 File storage	<= File capacity limit
	 Log storage	+	 Log storage	+	 Log storage	+	 Log storage	<= Log capacity limit
	Dataverse 1		Dataverse 2		Dataverse 3		Dataverse X	

Figure 3.2: Storage capacity limits

As we can see, the database storage capacity, the file storage capacity, and the log storage capacity from all the environments in a tenant are added together and compared against the capacity limits. If the database capacity limits are reached, no new environment can be created.

## Request limits and allocations

These limits apply to every individual Power Platform environment separately and limit the number of requests any individual user can make against the Power Platform Dataverse solution within a 24-hour period. Requests that count toward these limits can come from the following:

- Interactive end user work
- API calls against Dataverse
- Canvas apps or Power Automate flows communication with Dataverse using the Dataverse connector

The values of the request limits depend on the license type the user communicating with the platform has. These vary between 6,000 and 250,000 requests per 24-hour interval.



### Tip

Microsoft allows us to purchase capacity add-ons to increase these request limits.

## API limits

These limits apply to every individual Power Platform environment separately, but only for API requests. These are evaluated automatically in 5-minute windows for every web server within a scale group. The following measures are evaluated regarding the API limits:

- Number of requests
- Execution time
- Number of connections

These limits can have negative performance impacts on Power Platform integration solutions and specifically on the data migration process, which is usually performed as part of the solution's deployment.

### Tip



If you consult the Microsoft Power Platform documentation, you will see that there are recommendations on how to handle the signals coming from the platform in case the API limits are exceeded. Please refer to the following article: <https://docs.microsoft.com/en-us/power-apps/developer/data-platform/api-limits>.

For large data migration scenarios, there is a possibility to contact Microsoft support to lift the API limits during the migration period window.

Let's move on to the next section, where we'll learn about Power Platform data connectors.

## Power Platform data connectors

**Power Platform data connectors** are basically wrappers around certain APIs provided by various Microsoft and non-Microsoft services. These connectors allow us to connect to those APIs from **canvas apps**, **Power Automate flows**, and **Azure Logic apps** in a low-code/no-code fashion without bothering with the technology details of the APIs. There are three types of connectors:

- **Standard connectors** are not bound to any licensed technology and can be used with any subscription; for example, with a Microsoft 365 subscription.
- **Premium connectors** require a Power Platform subscription in order to be used.
- **Custom connectors** are connectors developed by a customer for connecting to a certain technology, for which there is no public connector available or the public connector capabilities are not sufficient for the implementation. Custom connectors are available only in the tenant of the customer and they are always considered premium.

Power Platform connectors offer the following capabilities for canvas apps and **Power Automate** flows:

- **The Tables capability** is available for certain connectors and can be used in canvas apps to connect the data source behind the connector with canvas app controls (gallery, data card, and so on). The Dataverse connector is an example of a connector that provides the *Tables* capability.
- **The Triggers capability** is available for certain connectors and can be used in Power Automate flows to start the execution of the flow. For the Dataverse connector, for example, there are triggers such as *Record created*, *Record updated*, *Record deleted*, and *Record selected* available. *Record selected* is an indirect trigger since the flow is triggered manually by the user from a model-driven application.
- **The Actions capability** is available for certain connectors and can be used in both canvas apps and Power Automate flows and represents basically executions of changes in data or other manipulations, performed by the connector from within a canvas app or Power Automate flow. For the Dataverse connector, for example, there are actions such as *Create record*, *Update record*, *Delete record*, *Get record*, and many others available.

## Data Loss Prevention policies

**Data Loss Prevention (DLP) policies** are connector-targeted policies used within an organization to protect organizational data from unintended exposure. For example, this could happen when a Power Automate flow reads some internal financial data from a database and submits it to social networks.

DLP policies can be created on two scope levels:

- **Tenant level:** This is valid across all or selected Power Platform environments in the tenant. Only a user with a tenant-level administration role can manage these policies.
- **Environment level:** This is valid only in the respective selected environment. These policies cannot override the tenant-level policies and can be managed by an environment administrator; no tenant-level administration role is required.

These policies define rules for which connectors can be used together in a solution to effectively block possible dangerous information exposure. The DLP policy can place the connectors into any of the following groups:

- **Business data:** Connectors in this group can be used only in combination with other connectors from this group, not with any connectors from the other groups. This group is initially empty and it is recommended to put all connectors to IT systems containing sensitive business data here.

- **Non-business data:** This is the default group that all the connectors are initially part of. When configuring the policy, connectors to IT systems not containing any business-relevant data are placed here.
- **Blocked:** Connectors in this group are entirely blocked within the specified scope and cannot be used in canvas apps and Power Automate flows.

The DLP policies allow even more detailed configuration, especially:

- Specification of which data connector **actions** are allowed, and which are blocked
- For certain selected connectors specification, which **endpoints** can be connected to, and which are blocked

By default, no DLP policies are created in a new Power Platform ecosystem. It is recommended to create proper policies as soon as possible, before the makers start building apps and flows.

It is important to understand that a combination of multiple DLP policies always leads to the **most restrictive result**. This implies that DLP policies with wider scope need to have less restrictive configuration, while policies for certain highly sensitive environments can enforce heavy restrictions.

## On-Premises Data Gateway

**On-Premises Data Gateway** is a specific software solution for hybrid scenarios, enabling the use of the organization's own on-premises data sources within the following Microsoft cloud services:

- Power Apps canvas apps
- Power Automate cloud flows
- Power Automate Desktop flows connected with cloud flows
- Power BI
- Azure services (Azure Logic Apps and Azure Analysis Services)

There are two different types of On-Premises Data Gateway:

- On-Premises Data Gateway: This can be shared among multiple users.
- On-Premises Data Gateway (personal mode): This can be used only for one user and only for Power BI.

An On-Premises Data Gateway technically consists of the following components:

- **On-Premises Data Gateway cloud service**
- **Azure Service Bus**
- **On-Premises Data Gateway local installation**

The architecture of the gateway is illustrated in the following diagram:

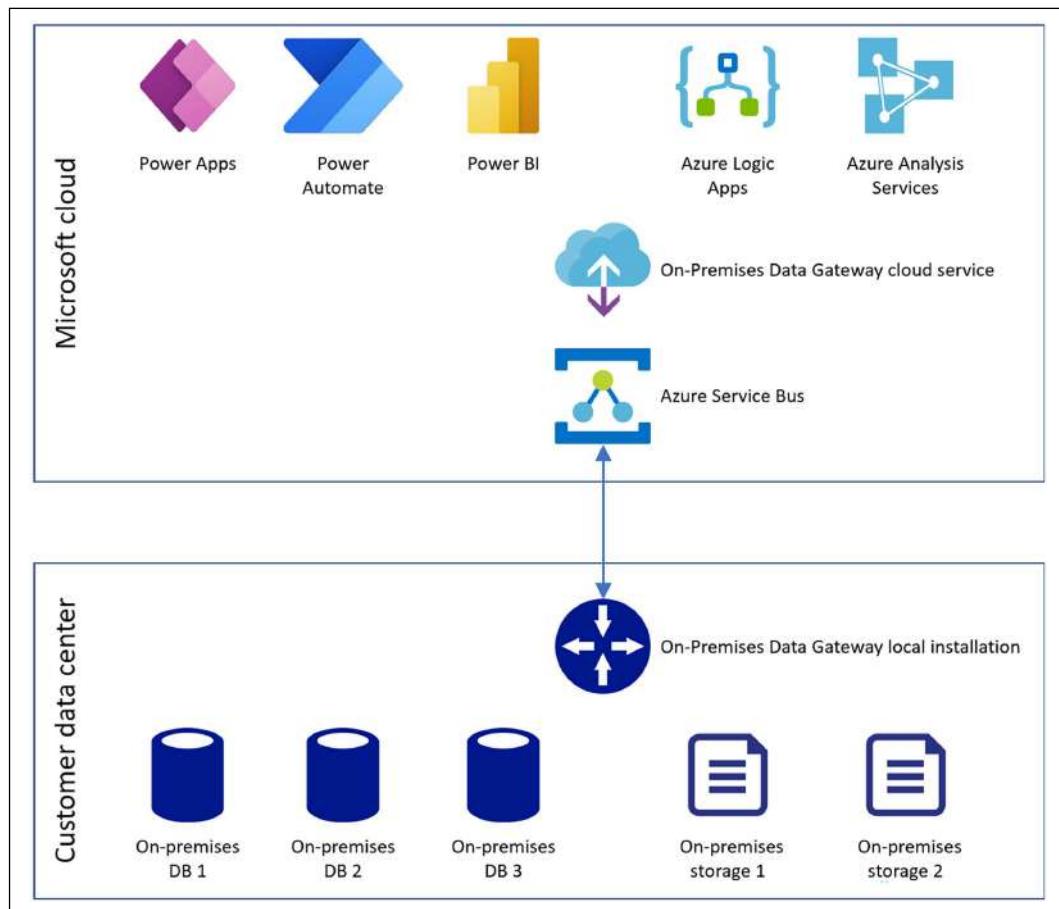


Figure 3.3: On-Premises Data Gateway architecture

The respective components of the gateway are responsible for encrypting and decrypting the on-premises data source's credentials and data, connecting to the data sources, routing the requests from the cloud and the responses from the on-premises data sources, and so on. The use of the **Azure Service Bus** component in the architecture provides a significant security benefit since the connection between the on-premises data center and the cloud is always outbound. This capability makes it possible to use an on-premises-to-cloud connection without the need to open any inbound communication, thus exposing the internal network to potential attacks guided by malicious players.

## Managed environments

Before we drive our attention to the structure of Power BI, there is a new feature, as of writing this book in preview, called **managed environments**. The purpose of these environments as compared with normal environments is to bring more governance, administrative control, and insight into the environment. Currently, the managed environments provide the following features:

- **Weekly digest:** An administrative overview of all managed environments, containing analytical information about apps, flows, active users, unused apps, most popular apps and flows, and more.
- **Sharing limits:** This feature makes it possible to define limits on how many users can be used at maximum when sharing apps or flows. In addition, sharing with security groups can be disabled entirely.
- **Data policies:** A consistent view of all data loss prevention policies, applying to a particular environment.

Managed environments are an interesting feature, moving toward better manageability and more control. At the same time, it is important to know that in a managed environment, every app and flow is considered **premium**, and every user needs to be equipped with a Power Platform license – a Microsoft 365 license or any other low-level license is not sufficient.

Next, we will look at the structure of Power BI, which is different from the Power Platform structure we've looked at so far.

## Learning about Power BI's structure

Power BI is technologically different from the other components of the Power Platform, and Power BI is not integrated into the concept of Power Platform environments. Power BI's structure consists of the following elements:

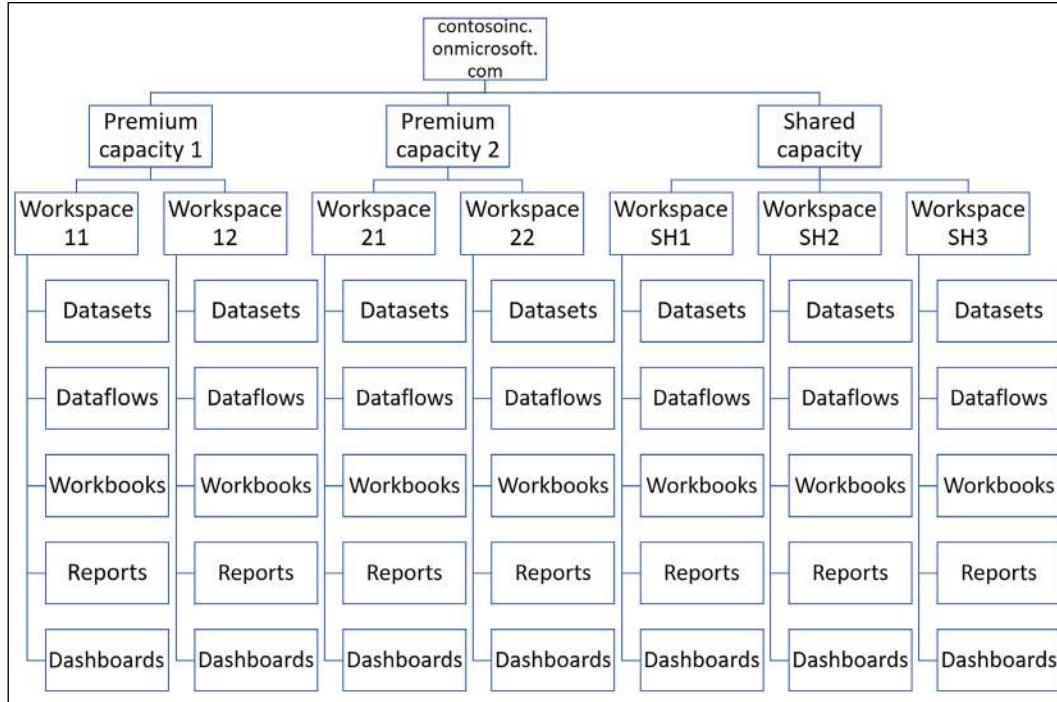


Figure 3.4: Contoso Inc. example structure of the different Power BI services

The different components in the Power BI hierarchy are used for the following purposes:

- **Capacity** is a Power BI concept that's used for a set of infrastructure resources (compute power) to run the Power BI service. There are two types of capacities – **shared**, where the resources are shared among multiple customers, and **dedicated**, where the resources are exclusively used by one customer. The dedicated capacity requires a **Power BI Premium** license.
- **Workspaces** are containers for other Power BI components (datasets, dataflows, workbooks, reports, and dashboards). Workspaces can be considered logically equivalent to Power Platform environments.
- **Datasets** are data collections used directly as data sources for Power BI reports.

- **Dataflows** are data sources prepared for pushing data into datasets.
- **Workbooks** are specific datasets based on Excel.
- **Reports** are the main visualization objects created in Power BI using data from the datasets.
- **Dashboards** are collections of tiles, widgets, and visualizations coming from multiple reports and other sources.

In this section, we learned about the Power Platform architecture. We also learned about the Microsoft cloud infrastructure and the customer cloud structure, with details about their central tenant services. We also learned about Power Platform technology and Power Platform environments, along with its components. Finally, we learned about the structure of Power BI.

In the next section, we'll understand the various types of clients available in Power Platform.

## **Understanding the Power Platform and Dynamics 365 clients**

Power Platform is an ecosystem consisting of several application types, and those application types offer several possible technical client types for desktop as well as for mobile workers. In this section, we will provide an overview of the various client types and the typical usage scenarios. This will allow you to design the best possible client configuration for various user groups of your Power Platform solution.

There are basically two types of clients: namely, **desktop** and **mobile** clients. We'll learn about them in the following sections.

### **Learning about desktop clients**

The Power Platform desktop clients are used on PCs, primarily running the Microsoft Windows operating system (OS), but some of the clients are also available for PCs with non-Microsoft OS. In the following sections, you will learn about possible desktop clients.

#### **Browser client**

The **browser client** is the most generic and easy-to-use client, within which all Power Platform applications can run. Currently, the following browser clients and operating system combinations are supported:

- **Microsoft Edge**
- **Google Chrome** (latest publicly released version on **Windows** and **macOS**)
- **Apple Safari** (latest publicly released version on **macOS** or **iPad**)

- Mozilla Firefox (latest publicly released version on Windows)

Using the browser clients for running Power Platform applications is the easiest way, since no client software installations are required.

## Dynamics 365 App for Outlook

**Dynamics 365 App for Outlook** is the modern successor of the legacy Dynamics 365 Client for Outlook. It provides the following features:

- Dynamics 365 App for Outlook runs within Outlook desktop and the Outlook web client on PCs, as well as in Outlook apps for Android and iOS on mobile devices. This is a significant improvement compared to the legacy client application.
- Dynamics 365 App for Outlook offers a simplified deployment with no local installation. Instead, once deployed as an **Office Add-in**, it integrates automatically with all the Outlook clients of the respective user.
- Dynamics 365 App for Outlook is a specific model-driven application and, as such, can be customized to reflect different customer requirements.

Dynamics 365 App for Outlook offers the following capabilities:

- View Dynamics 365 information about related contacts or leads within emails, appointments, and so on.
- Link emails, appointments, and so on with Dynamics 365 records.
- Add activities and create new Dynamics 365 records.
- Track contacts and more.

### Important note



All of the described Power Platform clients require the desktop versions of **Microsoft Word** and **Microsoft Excel** if you wish to use the template-based document generation with *Word*, or data export and import capabilities using *Excel* files.

## Unified Service Desk

The **Unified Service Desk (USD)** is a framework for building contact center and call center solutions supported by the Dataverse platform. The USD consists of two main components: a **Dataverse solution** and the **Agent Desktop application**.

These main components will be described in the next two sections.

## Dataverse solution

Dataverse is used to host the USD tables and offers full configuration capabilities for creating the agent desktop's composition, design, integrations, automations, and so on. For simple requirements, there is no need to use custom development; however, there are possibilities for deeper modification and extensibility using code.

## Agent Desktop application

The Agent Desktop application is a local desktop installable application that is used by contact center/call center agents for their daily business. The agent logs in to Agent Desktop, which then connects to the Dataverse environment from which the configuration for the business processes is then retrieved. This approach makes an on-time deployment of Agent Desktop on the agent's PCs possible. If the business functionality changes, it is changed centrally in the configuration of the Dataverse solution, without the need to reinstall anything on the agent's PCs.

The USD is based on the **User Interface Integration (UII)** framework, which, in turn, is a custom development framework containing the following capabilities:

- Integration and automation of applications
- Session management
- Hosting various application types (hosted controls, web applications, **Silverlight** applications, **Java applets**, **Microsoft Win32** applications, .NET applications, **Citrix** remote hosted applications, and so on)
- Automating communication across various channels
- **CTI integration** using the *CTI framework*

The USD is quite complex and setting up a comprehensive call center solution requires a lot of effort and appropriate expertise. Currently, there is a modern alternative solution built into the Dynamics 365 Customer Service module. The solution is a multi-session/multichannel client experience implemented in the following two Dynamics 365 applications:

- Dynamics 365 Customer Service workspace
- Omnichannel for Dynamics 365 Customer Service

Another modern alternative covering many of the USD capabilities is **Power Automate Desktop**, the robotic process automation component built into Power Automate.

When analyzing requirements and architecting solutions for client-side automations or multi-channel support, these technologies should be considered and could be potentially beneficial over the USD. Therefore, we will look at them in detail in the following two sections.

## **Omnichannel for Dynamics 365 Customer Service**

**Omnichannel for Dynamics 365 Customer Service** is a separately licensed add-on for Dynamics 365 Customer Service. It allows us to configure various additional communication channels for customer service agents. Currently, the following channels are available:

- Telephony channel
- SMS channel via **TeleSign** or **Twilio**
- Chat channel
- **Facebook** channel via **Facebook Messenger**
- **Teams** channel
- **WhatsApp** channel via **Twilio**
- **LINE** channel
- **Twitter Direct Message** channel
- **WeChat** channel
- APIs for building co-browse and remote assistance using third-party providers
- Integration of the telephony channel using the **Dynamics 365 Channel Integration Framework**

In addition to that, the module offers the already mentioned multi-session capability, automated task distribution to agents based on their available capacity or skills, and many more.

## **Robotic process automation with Power Automate Desktop flows**

Power Automate Desktop is the latest member of the Power Automate product family. Desktop flows make it possible to record manual interactive UI step-by-step activities, which need to be performed on legacy enterprise applications that don't have APIs. Those recordings are then used within Power Automate flows to automate the manual processes. This is done by playing the recordings back with real data that's been collected within the flow's automation from other sources.

The benefit of this capability is that there is no need to modify the existing legacy applications just for the purpose of integrating them into modern automation processes.

## Understanding mobile clients

In today's world, mobile clients for business applications have increased importance since the number of mobile workers is growing very fast. Power Platform provides mobile applications for all platform components for both Android and iOS free of charge. The following are the mobile clients that are available for these applications:

- **Dynamics 365 for Phones and Dynamics 365 for Tablets:** These are mobile clients for model-driven applications. Despite their names, they can run Dynamics 365 as well as non-Dynamics 365 model-driven applications. The user can select any of the apps they have access to and start it. The user interface's design and look and feel are identical to the design for the browser client, just optimized for the small form factor.
- **Power Apps mobile:** This is a mobile client for model-driven apps and canvas apps with similar capabilities as the previous one.
- **Field Service Mobile:** This is a specific model-driven mobile application, used for **Dynamics 365 Field Service**. The application supports optimized rendering for field service technicians, along with some additional features. Since it is also a model-driven app, it can be customized in the usual way as other model-driven applications.



**Important note**

The aforementioned mobile apps provide offline capability. This capability requires certain mobile offline configurations to be performed first.

- **Power Automate mobile:** Mobile client for Power Automate flows. Compared to the other mobile player apps, the Power Automate mobile app has extended capabilities, since the user can not only run the button flows, but also participate in approval processes, create their own flows, and monitor the whole environment with their own or shared flows.
- **Power BI app:** Mobile client for consuming Power BI dashboards the same way as they are consumed in the browser client.

In this section, we learned about the types of Power Platform clients; namely, desktop and mobile clients, along with their sub-types.

In the next section, we will focus on Power Platform administration and monitoring.

## Learning about Power Platform administration and monitoring

Administration and monitoring are very important aspects when it comes to using the Power Platform ecosystem in a company. The more complex and diverse the landscape and the greater variety of user groups who start using the Power Platform, the more important proper administration and monitoring is to achieve the necessary level of governance.

It is increasingly important, with the growing complexity of your own ecosystem, to ensure governance. You need to keep control and have the ability to take timely action in case of any disturbances, policy violations, and so on.

Power Platform can be administered and monitored in a multitude of ways. In this section, you will learn about all the different options available, from manual administration using the admin centers to fully automated possibilities using various tools the platform offers. You will also learn about different monitoring options, from built-in analytics to more sophisticated automated possibilities.

### Understanding Power Platform administration centers

Currently, there are four administration centers that are used to administer and monitor various components of the Power Platform. In the following sections, we will start by describing the administration and monitoring capabilities of those administration centers.

#### Microsoft Azure portal

The Microsoft Azure portal is a comprehensive portal, where you can administer all aspects of Microsoft Azure, but also certain Power Platform relevant areas. The portal can be found under the following unified URL: <https://portal.azure.com/>. From the Power Platform point of view, the most important administration capabilities are the following:

- User management
- Group management
- Assignment of licenses to users
- App registrations, used when developing an external application that will communicate with the Dataverse API
- Provisioning and management of Microsoft Azure services, used when building complex Power Platform solutions

Most of the mentioned capabilities can also be found in Microsoft 365 Admin Center.

## Microsoft 365 Admin Center

**Microsoft 365 Admin Center** is used for administration purposes regarding the Microsoft 365 centralized services, as described earlier in this chapter. The admin center provides a unified URL for every customer: <https://admin.microsoft.com>. You can perform the following Power Platform-relevant tasks in this admin center:

- Purchase Power Platform licenses and manage billing
- Create user accounts and assign them Power Platform licenses
- Create groups and assign members to groups

The admin center is also a starting point for all the other admin centers in the Microsoft 365 **Software as a Service (SaaS)** cloud services, like the Exchange, SharePoint, or Teams admin centers.

Microsoft 365 Admin Center provides a high-level monitoring overview of the platform and all the services included within Microsoft 365 and Power Platform. Monitoring analytics consist of the following areas:

- Reporting about the **productivity score** and **usage** of Microsoft 365 and some Power Platform services.
- **Service Health** contains the overall health status of all cloud services, service incident information, advisories, history of incidents, as well as management of issues.
- **Message center** provides important messages about upcoming platform and product feature updates, as well as the deprecation and decommissioning of services.
- **Connectivity overview of the Microsoft 365 services.**

For mobile administration, there is a mobile app version of **Microsoft 365 Admin** that offers a subset of administration and monitoring capabilities for Microsoft 365, including users, groups, and license management.

## Power Platform Admin Center

**Power Platform Admin Center** is the main administration center and is an entry point with links to other related administration centers in the Power Platform cloud. The admin center provides a unified URL for every customer: <https://admin.powerplatform.microsoft.com>.

The admin center provides the following administration and monitoring capabilities:

- **Environments management:** This capability encompasses features such as creating a new environment, configuring environment parameters and settings, managing Dynamics 365 apps, managing updates, managing solutions, managing backups and restores, managing environment copies, resetting environments, deleting environments, assigning Microsoft 365 groups to environments, and opening environments.
- **Analytics capabilities:** This capability provides a detailed analytical overview of many important environment- and application-related metrics. A more detailed description of this capability will be provided in the upcoming sections.
- **Management of support tickets.**
- **Management of data integration projects using Power Query.**
- **Management of On-Premises Data Gateways.**
- **Management of DLP policies.**
- **Tenant isolation:** This capability helps isolate your tenant from inbound connections from other tenants or block your tenant from connecting to other tenants.
- **Customer Lockbox:** This capability establishes a policy where access to your environments from Microsoft support personnel goes through an approval procedure.
- **Billing policies:** This capability is used for the pay-as-you-go Power Platform subscriptions, where you can decide to add the billing to the standard Azure billing.
- **Links to the other administration centers.**

Power Platform Admin Center contains comprehensive administration and monitoring analytics of all the areas of the platform, as described in the following sections. This capability does not contain any analytics of the business data contained in the applications; instead, it just serves administration and monitoring purposes.

## Capacity analytics

Capacity analytics reflects the new storage model (database, file, and log) and provides comprehensive analytics about the storage space of an environment equipped with the Dataverse database:

- Storage capacity by storage type, by source, and by environment.
- Detailed storage capacity at the individual environment level. This provides analytics about the top capacity resources that are consuming the most storage.

## Dataverse analytics

Dataverse analytics provides detailed insights into the following areas of the Dataverse structure:

- **Users** analytics (active users, their usage patterns over time, and so on)
- **Modes of access** analytics (active users by OS, browser, device type, business units, security roles, entities, and so on)
- **Table usage** analytics (most-used tables and custom tables)
- **System jobs, plug-ins, API usage**, and mailbox usage analytics

## Power Automate analytics

Power Automate analytics provides detailed insights into the following areas:

- **Flow run** analytics (daily, weekly, and monthly)
- **Flow usage**, created flows, flow errors, and sharing analytics
- **Connectors** analytics

## Power Apps analytics

Power Apps analytics provides detailed insights into Power Apps usage, location, errors, and performance analytics.

## Power BI admin portal

The **Power BI admin portal** has the following unified URL: <https://app.powerbi.com/admin-portal>.

This admin portal is used for Power BI administration and monitoring, including the following capabilities:

- Power BI tenant settings (overall settings for the whole Power BI environment within the tenant)
- Management of custom visuals and branding
- Management of dataflows (switching to your own Azure Data Lake storage instead of Power BI-provided storage) and more

Power BI monitoring analytics, which is contained within the Power BI admin center, provides detailed insights into usage metrics analytics (number of user and group datasets, reports, dashboards, top users and groups with the most dashboards, packages, reports, and so on).

## Understanding PowerShell administration and monitoring

PowerShell administration provides an automated way to administer cloud environments, and compared to the admin portals, it provides some more features that are not available in the portals.



### Important note

For those not familiar with PowerShell, it is recommended that you refer to an introductory learning resource such as <https://en.wikipedia.org/wiki/PowerShell>.

In this section, we will provide an overview of how PowerShell can administer and monitor various parts of the Power Platform.

## Microsoft 365 administration

There are two different PowerShell modules for administering Microsoft 365:

- **Azure Active Directory PowerShell for Graph** (module name: AzureAD)
- **Azure Active Directory Module for Windows PowerShell** (module name: MSol)

For the purpose of Power Platform, both modules provide administration capabilities for users, licenses, groups, and so on.

The following simple PowerShell code example illustrates how to create a new Azure Active Directory user using the AzureAD module:

```
$PasswordProfile = New-Object -TypeName Microsoft.Open.AzureAD.Model.PasswordProfile  
$PasswordProfile.Password = "ContosoUserPassword"  
New-AzureADUser -DisplayName "Contoso User 1" -PasswordProfile $PasswordProfile -UserPrincipalName "user1@contosoinc.onmicrosoft.com" -AccountEnabled $true -MailNickname "user1"
```

The preceding example demonstrates how to create a new Azure Active Directory user account with a password and nickname and enable this account at the same time.



### Important note

The Azure Active Directory Module for Windows PowerShell is planned for future deprecation once its capabilities are fully available in Azure Active Directory PowerShell for Graph.

These cmdlets provide an option to create PowerShell scripts such as the previous one and perform user, group, and license management in an automated way.

## Power Platform administration

There are multiple PowerShell modules for administering the Power Platform, as well as Dynamics 365. The following are the five most important modules for administration:

- The module for **Power Apps administrators** (module name: `Microsoft.PowerApps.Administration.PowerShell`) provides the following capabilities:
  - Dataverse administration
  - Management of canvas apps
  - Management of Power Automate flows
  - Management of connections
  - Management of custom connectors
  - Management of user settings
  - Management of DLP policies
- The module for **Power Apps app makers** (module name: `Microsoft.PowerApps.PowerShell`) provides the following capabilities:
  - Management of canvas apps
  - Management of Power Automate flows
  - Management of connections
  - The module for managing the **Power Apps checker service** (module name: `Microsoft.PowerApps.Checker.PowerShell`).
  - The module for connecting to Power Platform or Dynamics 365 environments and retrieving environment details (module name: `Microsoft.Xrm.Tooling.CrmConnector`)
  - The module for administering package deployments (module name: `Microsoft.Xrm.Tooling.PackageDeployment`)

The following simple PowerShell code example illustrates how to create a new Power Platform environment with the Dataverse database using the `Microsoft.Power_Apps.Administration.PowerShell` module:

```
New-AdminPowerAppEnvironment -DisplayName 'Contoso Production' -Location  
unitedstates -EnvironmentSku Production -ProvisionDatabase -CurrencyName  
'USD' -LanguageName 'EN'
```

This example demonstrates creating a new Power Platform environment with a specified display name in the United States region with an environment type of **Production**. It is specified that a Dataverse database will be created for this environment. For environments created with the Dataverse database, the primary currency and primary language are mandatory parameters.

This example illustrates one possible approach to administering Power Platform, by creating and using a set of PowerShell scripts for all the typical administration tasks you will encounter.

## Power BI administration

Microsoft Power BI also provides a set of useful PowerShell modules for all typical administration and monitoring tasks. The following PowerShell modules are available for Power BI:

- The **rollup** module, which is used to install all the other modules in a single installation step (module name: `MicrosoftPowerBIMgmt`).
- The **administration** module, which is used for encryption and auditing management (module name: `MicrosoftPowerBIMgmt.Admin`).
- The **capabilities** module, which is used for capacity management (module name: `MicrosoftPowerBIMgmt.Capacities`).
- The **data** module, which is used for managing datasets, dataflows, data sources, tables, columns, and rows (module name: `MicrosoftPowerBIMgmt.Data`).
- The **profile** module, which is used for logging in and out and executing calls to the Power BI REST API (module name: `MicrosoftPowerBIMgmt.Profile`).
- The **reports** module, which is used for managing reports, dashboards, tiles, exports, and imports (module name: `MicrosoftPowerBIMgmt.Reports`).
- The **workspaces** module, which is used for workspace management (module name: `MicrosoftPowerBIMgmt.Workspaces`).

The following simple PowerShell code example illustrates how to upload a new Power BI report file to the Power BI service using the `MicrosoftPowerBIMgmt.Reports` module:

```
New-PowerBIReport -Path '.\contososales1.pbix' -Name 'Contoso Sales  
Analysis Report' -Workspace ( Get-PowerBIWorkspace -Name 'Contoso Sales  
Workspace' )
```

This example demonstrates how to automatically upload a Power BI report package into a workspace specified by name.

As for the previous examples, we now know how to create a set of PowerShell modules to automate all the typical Power BI management tasks.

## PowerShell monitoring

The PowerShell modules that we mentioned earlier provide certain monitoring automation capabilities, such as the following:

- Collect Power Platform **environment usage** metrics (number of environments, apps, and flows)
- Collect Power Platform **connector usage** metrics (how many of a certain type of connector are being used by which apps and flows)
- Collect **On-Premises Data Gateway** metrics
- Write the collected monitoring information into the appropriate repository for reporting

When creating administration and monitoring automations with PowerShell, it is very important, besides other things, to get an overview of all the existing environments in the organization's tenant.

The following simple PowerShell code example illustrates how to generate a detailed list of all Power Platform environments containing the string Contoso in their display name, using the `Microsoft.PowerApps.Administration.PowerShell` module:

```
Get-AdminPowerAppEnvironment *Contoso*
```

This very simple example demonstrates how to use one of the monitoring PowerShell commands that are available. This can be a starting point for subsequent commands that perform administration tasks with the environment list.

## Learning about API administration

API administration provides another automated way to administer cloud environments, where a customer can develop the required administration procedures with code and integrate those procedures into their own overall management and administration IT system.

This approach is best suited for organizations that have their own centralized administration tool for administering their existing IT ecosystem and would, therefore, prefer to have the new Microsoft cloud environment managed the same way. In order to provide this capability, Microsoft cloud solutions are equipped with a standardized set of administration APIs. These are implemented using the widely used **REST** endpoint technology.

**Important note**

For those not familiar with the concept of APIs, it is recommended that you refer to an introductory learning resource such as [https://en.wikipedia.org/wiki/Application\\_programming\\_interface](https://en.wikipedia.org/wiki/Application_programming_interface).

In this section, we will provide an overview of how the available APIs can administer and monitor various parts of Power Platform.

## Microsoft 365 administration

**Microsoft Graph API** is used for Microsoft 365 administration with code. **Microsoft Graph API** has a rich set of programmability models for managing Office 365 and Windows 10, as well as **Enterprise Mobility and Security**. Microsoft Graph API provides a single REST-based endpoint at <https://graph.microsoft.com> for accessing all the resources in all Microsoft 365 products. For the purpose of Power Platform, the Microsoft Graph API provides administration capabilities for users, licenses, groups, and so on.

The following example demonstrates an HTTP request against the Microsoft Graph API for creating a new Azure Active Directory user:

```
POST https://graph.microsoft.com/v1.0/users
Content-type: application/json
{
  "accountEnabled": true,
  "displayName": "Contoso User 1",
  "mailNickname": "user1",
  "userPrincipalName": "user1@contosoinc.onmicrosoft.com",
  "passwordProfile" : {
    "forceChangePasswordNextSignIn": true,
    "password": "ContosoUserPassword"
  }
}
```

The preceding example demonstrates how to create a new Azure Active Directory user account with the same attributes as in the PowerShell example earlier in this section.

## **Power Platform administration**

There is a new API for Power Platform administration called **Microsoft Power Platform API** available, as a replacement for the deprecated **Online Management API for Microsoft Dataverse**. As of writing this book, the new API is in preview, with relatively limited capabilities, but plans are to cover the following administration areas:

- Billing and licensing management
- Applications management
- Environments management
- Portals management
- Governance management

The API is also of type **REST** and uses a unified URL: <https://api.powerplatform.com>. The API uses a concept of namespaces, representing the respective administration areas.

## **Power BI administration**

The **Power BI API** allows us to automate certain Power BI processes such as performing management tasks, pushing data into Power BI datasets, automatically refreshing datasets, and so on. Power BI provides the following API technologies:

- **Power BI REST API**
- **Power BI .NET API**
- **Power BI JavaScript API**

The APIs described in this section can support those who wish to build a management solution or extend an existing management solution, as illustrated in the following diagram:

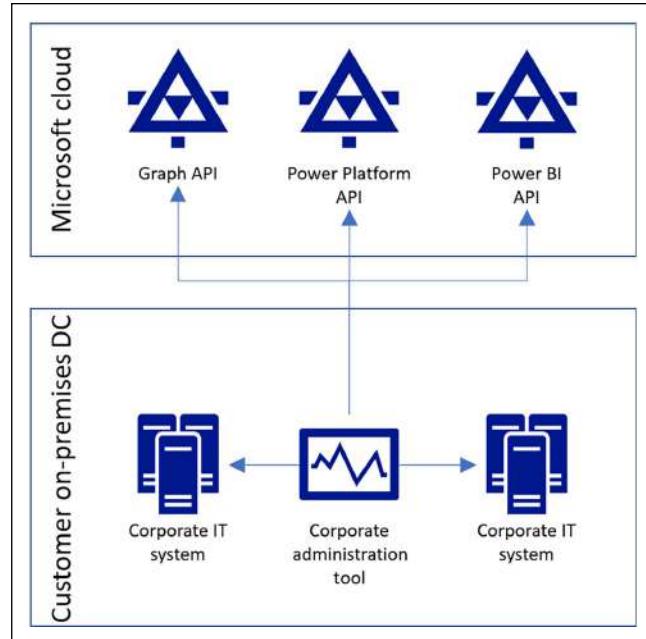


Figure 3.5: Centralized API-based management solution

Using a solution similar to the preceding one would greatly simplify and consolidate the administration efforts needed to manage the cloud environment in the following areas:

- Provisioning a new user account in the corporate network can be extended by provisioning an **Azure Active Directory account** for the user at the same time. This can be done using the **Graph API**.
- Granting permissions to corporate IT systems to users can be extended to granting such permissions to **Power Platform solutions** at the same time. This can be done by using the **Graph API** for assigning licenses.
- Administration of corporate IT systems can be extended with the administration of **Power Platform cloud environments** and the **Power BI instance**. This can be done by using the **Power Platform API** and **Power BI API**.



### Important note

The example solution provided here does not cover a possible integration between on-premises Active Directory and Azure Active Directory. You will learn about this integration in more detail in *Chapter 7, Microsoft Power Platform Security*.

In the next section, we'll look at administering and monitoring using **Power Automate**.

## Administration and monitoring using Power Automate

Since Power Automate is an automation solution, it can be used to manage and administer the Power Platform itself as well. For this purpose, Power Automate provides a set of **management connectors**, described as follows:

- **Microsoft Forms connector**, which can be used as part of an administration logic to collect manual input needed to perform the respective administration tasks
- **Approvals connector**, which can be used as part of an administration logic for requesting approvals within certain processes
- **Office 365 Users connector**, which can be used for searching for users, retrieving users' information, and updating the current user's details
- **Azure Active Directory connector**, which can be used for user and group management
- **Power Platform for Admins connector**, which can be used for retrieving, creating, updating, and deleting environments, as well as managing tenant and environment DLP policies
- **Power Apps for Admins connector**, which can be used for retrieving information about Power Apps and managing permissions
- **Power Apps for App Makers connector**, which can be used for retrieving information about Power Apps, managing permissions, apps, connectors, and connections
- **Power Automate Management connector**, which can be used for managing flows, connectors, and access rights
- **Power BI connector**, which can be used for adding rows to a dataset, refreshing a dataset, and more

Using the management connectors listed here, the customer can easily build their own Power Platform administration flows without using any code. The following screenshot illustrates a possible Power Automate flow for creating a new Azure Active Directory user:

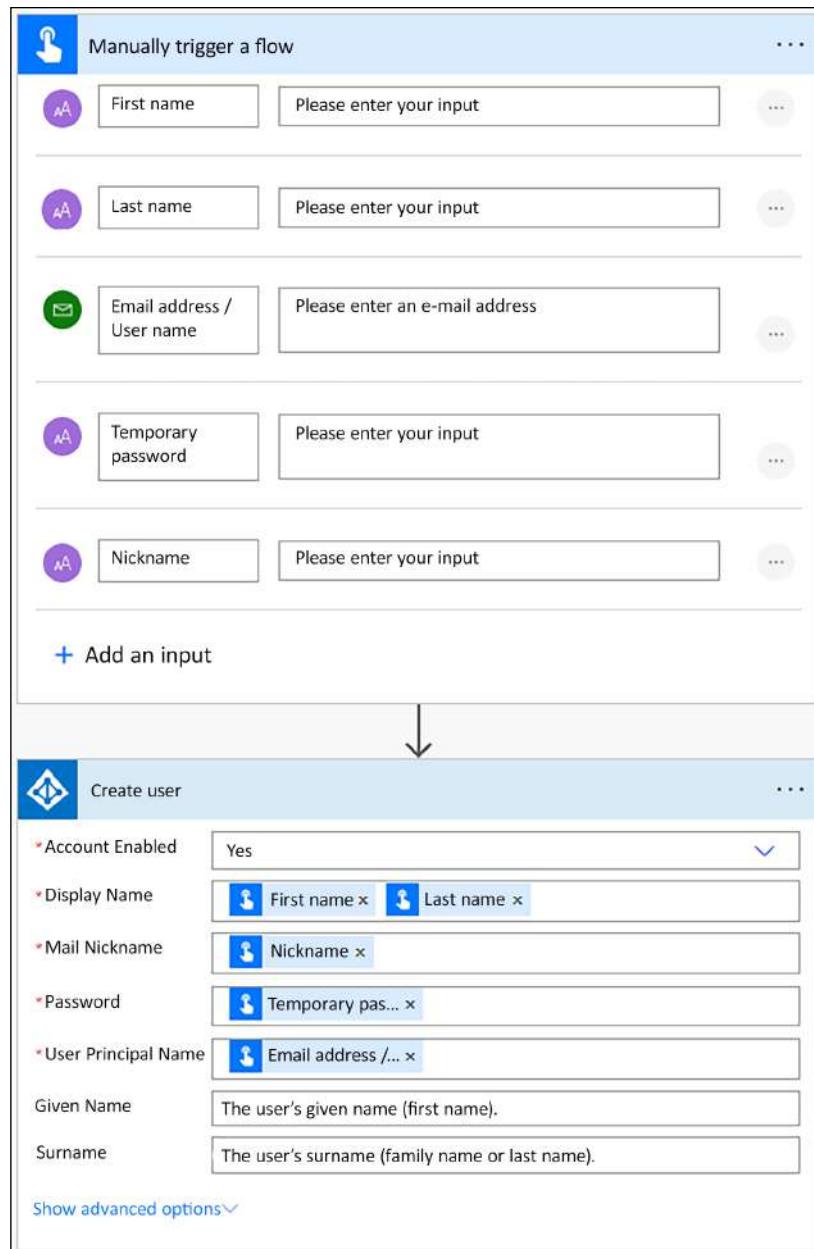


Figure 3.6: Power Automate flow for creating an Azure Active Directory user

As illustrated in the preceding screenshot, this is a manually triggered button flow, which collects basic information about the newly created user and then performs user creation in the Azure Active Directory using the Azure AD management connector.

The management connectors can be also used for certain monitoring tasks:

- **Administrators alerting and notifications:** This capability can be used to notify administrators about Power Platform ecosystem events (environment creations, app creations, flow creations, connector usage, custom connector creations, and so on).
- **Makers alerting and notifications:** This capability can be used to send welcome messages to new Power Apps or Power Automate makers, notify them about improper use of resources and suggested correction steps, and so on.

## Administration using Azure DevOps

Although Azure DevOps's primary purpose is solution development, it can be used for certain administration tasks as well. It is highly recommended to install the **Power Platform build tools** into Azure DevOps for every Power Platform implementation.

### Important note



You will find more details about using Azure DevOps for Power Platform projects in the next few chapters of this book, where we will discuss tools and techniques, application life cycle management, the implementation approach, methodologies, and so on.

The build tools provide the following administration capabilities:

- **Environment** management (create, delete, back up, and copy environments)
- **Solution** management (export, import, pack, and unpack deploy solution packages, as well as publish the customization)

## Learning about platform auditing

Another very important and useful monitoring tool is platform auditing, which is used for collecting detailed information and providing insights into what is happening within the various Power Platform applications. In Power Platform, there are two main auditing technologies:

- **Office 365 Activity Logging**
- **Dataverse auditing**

The capabilities of these two auditing technologies will be described in the following sections.

## Office 365 Activity Logging

**Office 365 Activity Logging** is an auditing feature that is part of the Office 365 Security and Compliance Center: <https://protection.office.com>.

**Office 365 Security and Compliance Center** is a comprehensive tool used for managing security and compliance in Microsoft Software as a Service cloud products. You will learn more about this tool in *Chapter 7, Microsoft Power Platform Security*.

The activity logging feature enables auditing of all Office 365 cloud services, including the Power Platform components. The logging needs to be enabled first. After you've enabled it, it will automatically start collecting auditing data. This feature collects data about the following events:

- **User management** events (adding, updating, and deleting users, as well as user password and license-related events).
- **Group management** events (adding, updating, and deleting groups, as well as group membership events).
- **Power Apps** events (creating, editing, deleting, publishing, and launching apps, as well as app permission changes).
- **Power Automate** events (creating, editing, and deleting flows, as well as flow permission changes).
- **Model-driven apps and Dynamics 365** events (administration events, creating, reading, updating, and deleting data in Dataverse tables, exporting to Excel, SDK calls, working with reports, and so on).
- **Power BI** events (creating, updating, deleting, viewing, publishing, retrieving, exporting, and so on of all Power BI elements).

All the data that's collected from these sources and their events can be filtered, sorted, and viewed in the Office 365 Security and Compliance Center. The data is stored on the auditing platform for a certain period of time, for up to 1 year. After this retention period, the data is automatically deleted.

To ensure longer retention and enhance the reporting and analytical capabilities provided, it is possible to export the data into a separate repository and process it with an appropriate tool.

In order to achieve this, the following steps need to be implemented:

1. Use PowerShell or Power Automate flows with the management connectors.
2. Connect to the **Office 365 Management Activity API**: <https://manage.office.com>.
3. Extract the required data from the activity logging repository.

4. Store the data in a permanent repository. The best option for further analytics and reporting would be a relational database, such as Azure SQL.

Next, we'll look at Dataverse auditing.

## Dataverse auditing

Another auditing option is to use traditional Dataverse auditing, which stores the auditing data in a specific auditing area (log storage type) in the Dataverse database. Auditing only works within the boundaries of a Dataverse database and the amount of information that's tracked is smaller; for example, no read transactions can be audited. This auditing capability needs to be first configured in the Power Platform Admin Center before it can start collecting auditing data.

## Understanding application monitoring

Another important monitoring option is to monitor the Power Platform applications directly in order to gain an analytical overview of the usage of various parts of the solution, including performance metrics, errors, and so on. A recommended way to build centralized application monitoring is to make use of Azure Monitor's capabilities, specifically **Azure Application Insights** and **Azure Log Analytics**.

Azure Application Insights is an Azure Monitor feature for monitoring application performance.

Azure Log Analytics is another Azure Monitor feature to analyze log files, generated by various source systems.

**Model-driven apps** integration with **Application Insights** needs to be implemented with code. The following integration possibilities are available:

- **Client-side integration** using JavaScript code can be used in any place in a model-driven application that supports client-side event handling.
- **Server-side integration** using Azure Functions and server-side registered **webhooks** event handlers or plug-in error handlers can be used for any server-side events.
- **Canvas apps** integration with Application Insights can be easily implemented by configuring the Application Insights **instrumentation key** in the canvas app settings.
- Integration with **Log Analytics** can be implemented using the **Azure Log Analytics Data Collector** connector from any canvas app or Power Automate flow.
- Another possibility is to extract data about previous Power Automate runs from the platform using PowerShell, and then write that data into Log Analytics using code with the **HTTP Data Collector API**.

In this section, we have learned about the Power Platform administration and monitoring aspects. We also learned about administration centers, the Power BI admin portal, and API administration. We also gained knowledge about administration and monitoring using *Power Automate* and *Azure DevOps*.

With this, we have concluded the overview of the Power Platform architecture and move now into a different area, where we are going to look into architectural best practices.

## Presenting architectural best practices

Creating a robust optimized enterprise architecture using *Microsoft Power Platform* is not easy and requires a lot of deep technical understanding and practical experience. In this section, you will become familiar with some of the best practices that are useful for setting up the Power Platform ecosystem.

### Introducing single tenants or multiple tenants

As explained earlier in this chapter, an Azure Active Directory tenant is the highest node in the customer's cloud ecosystem. Usually, when a new customer purchases a Microsoft cloud service, they obtain their new tenant as part of their subscription. When purchasing additional Microsoft cloud services, those are added to the tenant, which was created with the first purchase. But there are situations where a single tenant is not sufficient for a customer, and they need additional tenants. This is specifically the case when they plan to implement certain on-premises Active Directory integrations with the cloud. The following could be possible reasons for additional tenants:

- Separate development and testing Azure Active Directory environments are required.
- The customer has a complex enterprise ecosystem with a very heterogenous Active Directory structure.
- The customer requires Active Directory integration using a topology that is not supported for single-tenant integration.

In the following sections, we will explain the options for using multi-tenant environments.

### Development and testing environments

Large customers might require a permanent development and/or testing environment for Active Directory, specifically to develop, test, and maintain the Active Directory integrations with the cloud. They might want to develop and test specific security and data protection solutions before they go into production on the main production tenant. In those situations, it is possible for a single customer to purchase multiple Azure Active Directory tenants from Microsoft using special contractual agreements.

Technically, this would be a set of independent integrations between multiple on-premises active directories and multiple Azure Active Directory tenants, with a separate set of user accounts in every such integration, as shown in the following diagram:

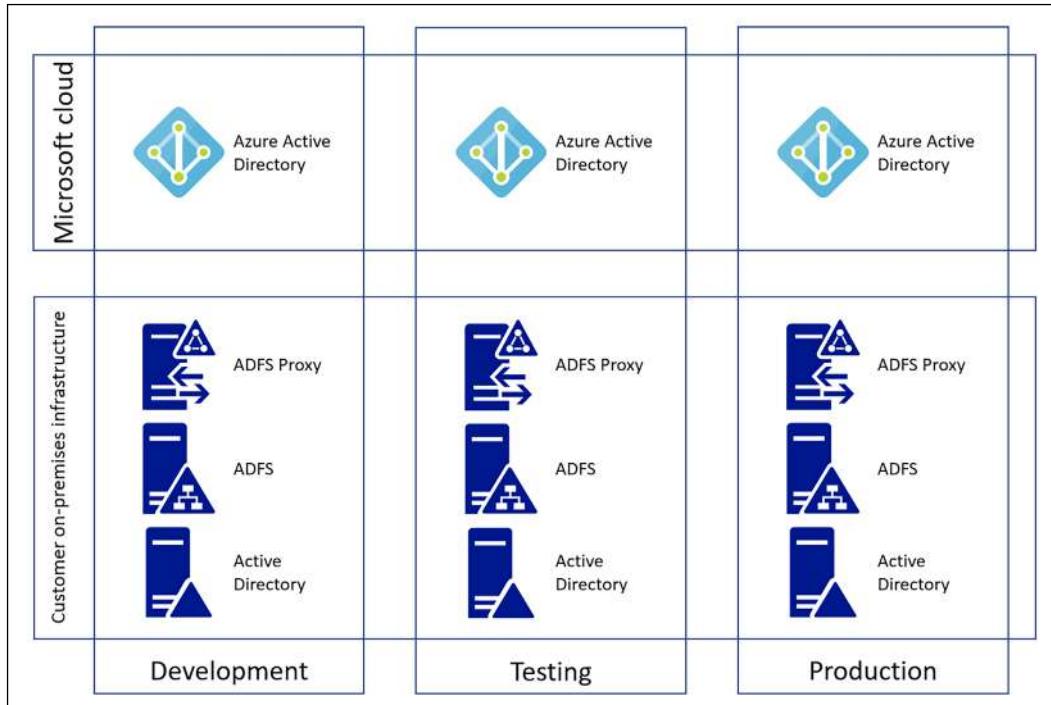


Figure 3.7: Multi-tenant environment

When using this multi-tenant approach, any cloud services such as Microsoft Power Platform are usually deployed **only in the production tenant**. If so, the above setup is free from any typical multi-tenant issues like multiple not-synchronized user accounts, inability to perform centralized cross-tenant administration, and so on.

## Unsupported integration topology

Another, more complex situation occurs when, for whatever reason, the complexity of the customer's Active Directory environment is not supported for integrating with a single Azure Active Directory tenant. Azure Active Directory supports many different topologies, as described in the following product documentation: <https://docs.microsoft.com/en-us/azure/active-directory/hybrid/plan-connect-topologies>.

However, certain topologies are not supported, specifically the following:

- Using several Active Directory synchronization services against one single Azure Active Directory domain
- Any configuration where the same Active Directory object (user account, and so on) would be synchronized into multiple Azure Active Directory tenants
- Any cross-domain synchronizations between multiple Active Directory forests and Azure Active Directory tenants

These complex situations might dictate the use of multiple tenants for **production use**. Distributing any cloud solutions – and Power Platform applications are no exception – across multiple tenants for production purposes imposes some serious issues that need to be taken into account:

- Within the same tenant, accessing different Power Platform environments with the same users is easy and is just a matter of security settings. However, Power Platform environments in separate tenants are totally separated and cannot be accessed with the same user credentials. A user would need separate and non-integrated credentials to be created in every tenant.
- While, within the same tenant, certain administration tasks are easy to perform (for example, copying one environment into another), for environments in separate tenants, this is not possible.
- Data integration for environments in the same tenant is much easier to achieve than it is for environments in separate tenants, simply due to the need for managing record ownership over separated and non-integrable user groups.

Generally, the best practice is to **avoid using more than one tenant** when establishing Power Platform solutions in an organization. If there is a need to have multiple Power Platform environments for production purposes, the preferred way is to have them all in the same tenant. Based on your requirements, the environments could be provisioned in the appropriate cloud regions for better performance, data residency, and other reasons.

If that still does not work and a multi-tenant topology is inevitable, then for Power Platform solutions, the following options are possible:

- Central consolidation environment
- Central reporting environment

In the following sections, we will describe the possibilities of these two options.

## Central consolidation environment

For this option, a central consolidation Power Platform environment would need to be created, along with custom data integration solutions, in order to consolidate all or selected data in the central environment, as illustrated in the following diagram:

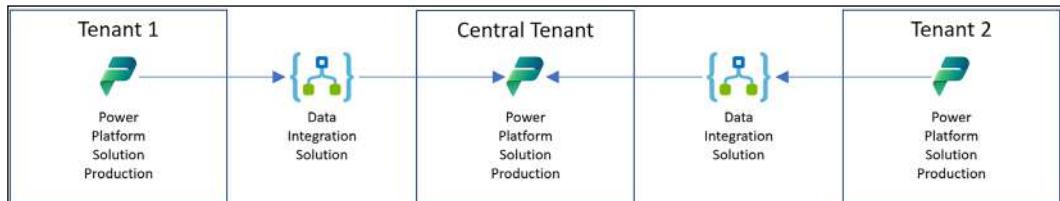


Figure 3.8: Central consolidation environment

The example approach shown in the preceding diagram consists of three tenants and in every tenant, there is a Power Platform environment. In order to achieve a data consolidation in the central tenant, we need a custom data integration solution from the two satellite tenants into the central tenant.

This solution will work under the following circumstances:

- The user groups in all the tenants are *separated and non-integrable*, where the satellite tenants might represent regional subsidiaries and the central tenant user group represents the headquarter management and central services.
- The data integration solutions consolidate the data from the tenant-based solutions into the central solution, mainly for *read-only* purposes. Any data modifications in the central tenant would not be replicated in the solutions in the satellite tenants.
- The data integration solutions would need to perform an appropriate *user mapping*, since the user groups are separated and non-integrable and there is a need to establish ownership for every record in the Power Platform solution in the central tenant.

A more complex version of this approach would be if the requirements were to dictate a bi-directional data integration, but fortunately, this is not typical.

## Central reporting environment

Another consolidation solution would need to implement a centralized reporting component based on Power BI, along with custom data integration solutions, to consolidate all or selected data in Power BI, as illustrated in the following diagram:



Figure 3.9: Central reporting environment

The example approach shown in the preceding diagram consists of three tenants. In the two satellite tenants, there are Power Platform environments, while in the central tenant, there's a Power BI solution. In order to achieve data consolidation in the central tenant, a custom data integration solution is needed from the two satellite tenants into the central tenant.

This solution will work under the following circumstances:

- The user groups in all the tenants are *separated and non-integrable*, where the satellite tenants might represent regional subsidiaries and the central tenant user group represents the headquarter management and central services.
- The data integration solutions *do not need to perform user mapping*, since Power BI has a different security concept, and keeping the original record ownership information can be beneficial for analytical and reporting purposes.

The above situation with the two possible solution approaches is not uncommon in large organizations, where multi-tenancy is just a result of a homegrown development. The experiences from multiple large implementation projects show clear benefits of the latter – Power BI-based solution – approach, which might be considered best practice for this situation.

## Understanding environment strategies

Designing an environment strategy for complex and long Power Platform implementation projects is not trivial and requires a lot of parameters to be considered. In this section, we will present the best practices for setting up an environment ecosystem for developing and operating Power Platform solutions.

Earlier in this chapter, we provided an overview of all possible Power Platform environment types. But for enterprise projects, only the following two are real candidates to be used when setting up an environment strategy:

- Sandbox
- Production



### Important note

In the following sections, you will learn about the best practices for various environment compositions. Please keep in mind that every environment in the presented compositions that is not labeled *Production* is of the *Sandbox* type.

These best practices are generally valid for any Power Platform solution, but specifically for Dataverse-based solutions, due to the additional complexity Dataverse brings to the equation.

## Default environment

The default environment plays a specific role in the ecosystem since this environment exists always as a single environment instance and cannot be removed. The environment is used behind the scenes, for example, when creating apps directly in SharePoint or in the context of Project Online.

The default environment is missing an access control; basically every licensed user has access to it. This limits the usage scenarios to just general playground and developing, not business-critical simple apps and flows with low impact on the organization.

## Developer environment

The developer environment is a specific type of environment with a certain usage limitation, which is that only the creator/subscriber for the developer plan can access it. The use of this type of environment in an environment strategy is limited to just an individual playground, where a developer can prove certain concepts, or develop individual solution components. It is not recommended to incorporate this type of environment into an environment setup for large Power Platform projects.

## Shared test and production environment

This environment strategy can be used in smaller and less business-critical solutions, where sharing with other solutions does not present any risk. The environment setup is presented in the following diagram:

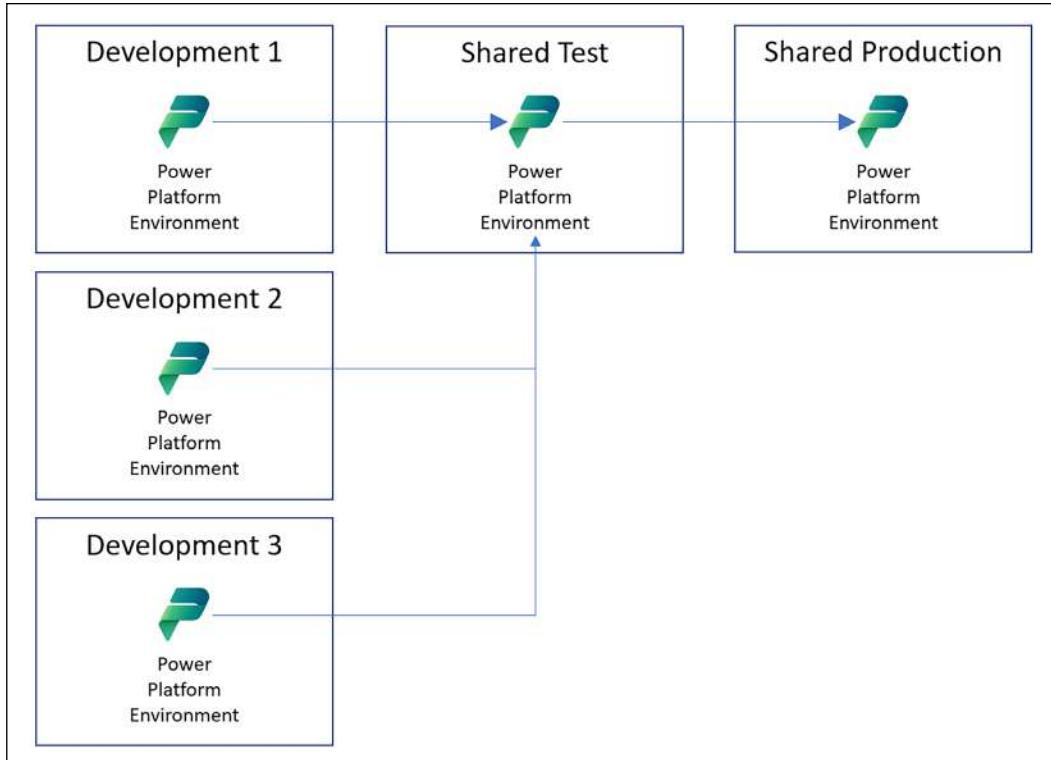


Figure 3.10: Environment setup for shared test and production

As you can see in the diagram, there are multiple independent development environments, where various teams develop smaller solutions. Once the solutions are ready to be tested, they move to the shared test environment. Later, when finished, they move to the shared production environment, where they operated together with the other parallel solutions. This strategy requires proper administration, security setup, and governance, specifically around the security setup in Dataverse and regarding the proper DLP policies specified to ensure a balance between security requirements and usability.

## Dedicated environment

Dedicated environments are usually used for complex, business-critical solutions. The word *dedicated* means that there is an environment setup exclusively for the single solution. The simplest dedicated environment setup is presented in the following diagram:

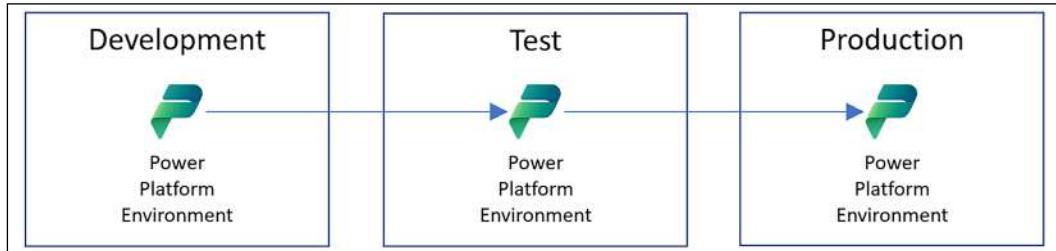


Figure 3.11: Environment setup for dedicated environments

As you can see in the diagram, the simplest setup contains the development, test, and production environments, all used for the single business-critical solution.

These environments are used for the following purposes:

- **Development:** This environment is used by the development team to develop the solution, including unit testing.
- **Test:** This environment is used for all testing except unit testing, such as system integration testing or user acceptance testing. In the case of any integrations, this environment is integrated with the customer's test versions of their IT solutions.
- **Production:** This environment is used for production purposes and is integrated with the production versions of the customer's IT systems and solutions.

A dedicated environment setup can be much more complex than the presented one, depending on the complexity of the solution, the development teams, the testing requirements, and more. Let us analyze the different factors mentioned and present some best practices.

## Complex solution/complex team structures

If the developed solution is very complex, or the team structure is very diverse, it might be necessary to have more than one development environment in the setup. Examples of these situations can be:

- The solution requires significant custom development, where the developers need dedicated development environments to be able to develop and at the same time not disturb or be disturbed by the ongoing standard customizations.

- The complex solution is broken down into sub-projects consisting of separate, independently-developed workloads, for example, basic customer management, sales management, customer service management, etc. In this situation, multiple development environments can also be necessary.
- There are distinct teams developing the solution, possibly from different companies, again requiring separate environments.

In all these situations, it is necessary to be able to manage the solutions coming from the various environments and consolidate them into a consistent solution package being able to be transferred to the downstream environments. This can be achieved either by using a development master or by properly handling the solutions in a source control system, like Azure DevOps.

## Complex testing

For customers that require several stages of testing, an appropriate testing environments cascade can be implemented:



Figure 3.12: Environment setup for complex testing scenarios

In the preceding diagram, there is a three-stage testing cascade of **system integration testing (SIT)**, **user acceptance testing (UAT)**, and **operational acceptance testing (OAT)**, which is a typical required setup for some large customers. The purposes of these environments are as follows:

- **SIT:** This environment is used for technical testing of the solution as a whole, including all possible integrations with other IT systems.
- **UAT:** This environment is used for final end-to-end functional testing provided by human testers.
- **OAT:** This type of testing is used to verify the operational readiness of the solution to be supported so it can become part of the production environment.

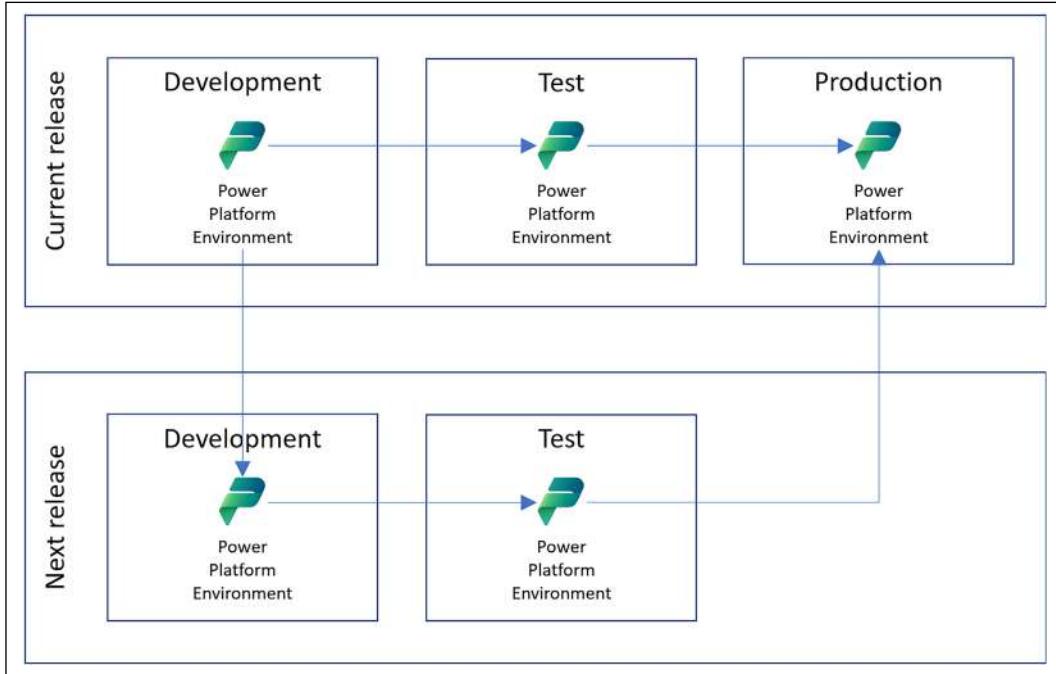
## Multiple release strategy

In the case of a large and complex Power Platform solution, where multiple solution releases are planned, it is necessary to ensure that there are two development and testing streams:

- **Main development stream:** For developing the next major solution release (version “N+1”)

- **Support development stream:** For bug fixing and minor improvement purposes for the existing solution's release in production (version "N")

This setup can be seen in the following example diagram:



*Figure 3.13: Environment setup for the multiple release strategy*

While, in the **Current release**, the development and test environments are used to resolve any possible issues and bug fixes, in the **Next release** the development and test environments are used to develop the next solution version that will be released in the future.

There must be a structured way to properly transport all the identified bugs and other issues from the support stream into the main development stream. This helps to ensure that those issues will not be replicated in the next major solution release. There is no best way to do this; it is usually managed using a bug-tracking system.

## Product upgrades

In the case of a longer-running solution implementation, where a new Power Platform product release is expected during the implementation, it is necessary to ensure that a separate environment for development and testing is established.

This helps verify the full compatibility of the developed solution with the new Power Platform product release. For those dedicated environments, the preview feature must be activated at the beginning of the preview window (between the preview's availability and the new product's release date) in order to ensure enough time for all necessary tests.

This setup is shown in the following example diagram:

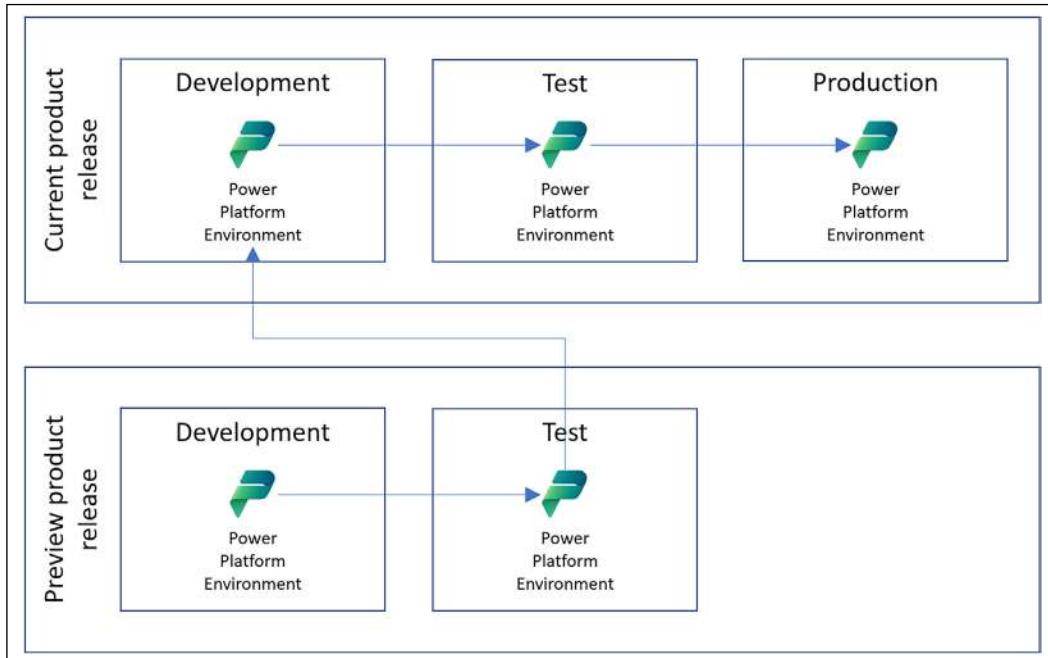


Figure 3.14: Environment setup for product upgrades testing

The **Current product release** is used to develop the solution on the Power Platform's major product release, also called **general availability (GA)**. The **Preview product release** must have the Power Platform product preview feature activated. It is used to test the current solution on the potential compatibility issues of the solution with the next Power Platform product release.

For this setup, there must also be a way to properly transport all the identified bugs and other issues from the product preview environment into the main development environment. A proper bug-tracking solution can be used to track these findings and provide input to the development team.

## Other environment types

According to specific needs, other environments can be specified and provisioned to fulfill customer- or project-related requirements, such as the following:

- **Training environment:** This environment can be used to train the users. It can be created as a copy of the production environment, which contains real data, to make the training experience realistic.
- **Feature testing environment:** This environment can be used to test specific solutions from AppSource or other sources to decide on the feasibility of the solution. It can be created as a trial environment. This is because trials are free of charge, do not count toward the storage limits, and have a limited lifetime, which is usually 30 days.
- **Developer sandbox environment:** This environment can be used outside of the solution implementation as a developer playground for verifying certain developer concepts. For this purpose, the developer type of environment can be used since these environments are free of charge, do not count toward the storage limits, and are permanent, which means the developer can use them for a longer period of time.

## Environment regions

As you learned earlier in this chapter, every environment can be created in a selected Power Platform region, independently of the region of the tenant. This is a very useful feature, making it possible to fulfill specific requirements:

- **Data sovereignty requirements:** In certain countries, there might be a legal requirement to store data of some types only within the borders of that country.
- **Performance requirements:** The distance between the users and the location of the Microsoft data centers influences the overall performance of the Power Platform solution. This is due to the latency time between the two points. In order to optimize the performance, it is useful to select a Power Platform region near to the majority of the users of the solution.

While it is possible to specify the best regional environments configuration using multiple production environments in various regions, it is also necessary to keep the possible **data consolidation consequences** in mind, since there is no out-of-the-box solution for automated data synchronization between Power Platform environments.

**Tip**

Consider using more than one Power Platform production environment only when absolutely necessary to avoid the need to build a custom data consolidation solution.

## Administration and monitoring

The Power Platform is being designed more and more for **citizen developers** and **power users**. The goal is to break the barriers of large and long-running IT projects and empower users to bring business value fast by letting them develop small solutions in a low-code/no-code fashion. This is certainly a great idea, but there are also some risks with the uncontrolled distribution of app maker rights into an organization. We need to consider specifically the following risks:

- Governance over environment creation
- Security risks connected with *shadow IT* (uncontrolled creation of IT components by non-IT personnel)
- Data protection risks regarding the unrestricted use of connectors
- Maintenance of the apps and flows created

There are various approaches, tools, and methods we can use to mitigate these risks, some of which were already described earlier in this chapter, such as the proper use of DLP policies or using monitoring tools. In order to support customers so that they can achieve proper governance, Microsoft offers a very useful **Center of Excellence Starter Kit** for Power Platform: <https://aka.ms/COEStarterKit>.

The kit consists of a collection of Dataverse solutions, model-driven apps, canvas apps, Power Automate flows, connectors, Power BI report, templates, and so on. These are designed to help customers with large and growing Power Platform ecosystems in the following areas:

- Managing an inventory of all Power Platform tenant resources
- Managing DLP policies
- Managing the auditing of apps or flows
- Analyzing connector usage
- Managing unused apps
- Onboarding app makers

It is highly recommended to adopt the starter kit since it significantly accelerates the process of establishing overall Power Platform governance in a complex ecosystem.

The following screenshot shows an example of one of the many Power BI dashboards contained in the CoE Starter Kit, which is providing an environments overview:

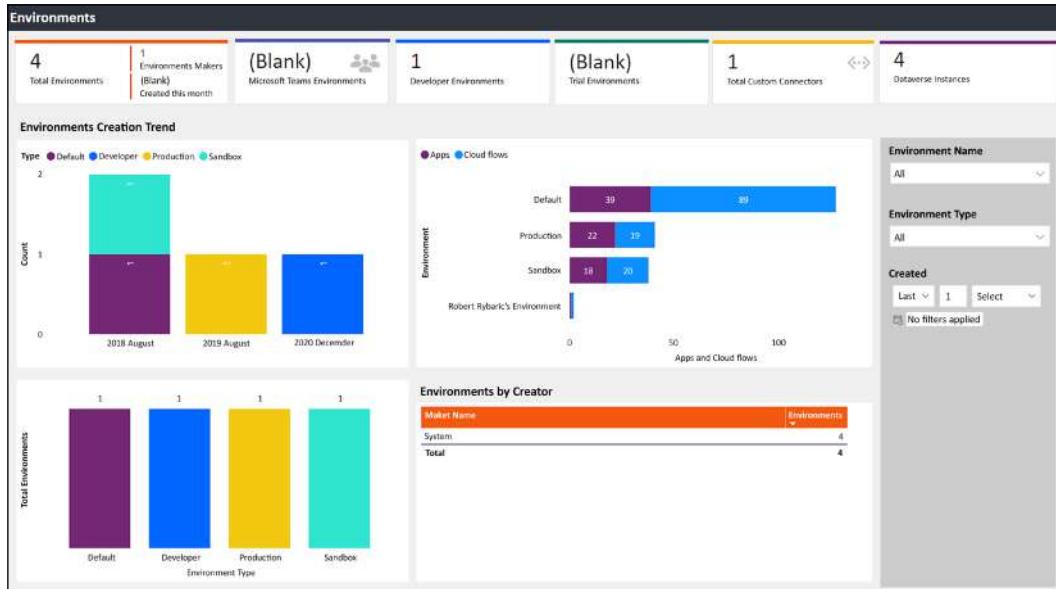


Figure 3.15: CoE Starter Kit – Environments overview

The dashboard presented in the preceding screenshot is providing a detailed overview of environment creations by type on a timeline, the number of apps and flows per environment, the most active app creators, and more. The preceding example illustrates one small part of the offering, but the whole **CoE Starter Kit** is much more comprehensive and provides real help for organizations planning large Power Platform implementations.

**Important note**

In order to specifically support automation governance when using Power Automate, Microsoft is releasing a new starter kit called Microsoft Power Platform Automation CoE Starter Kit. As of writing this book, the starter kit is in preview. The kit has a similar structure and purpose as the Power Platform CoE Starter Kit, but strongly focuses on all aspects of the Power Automate product.

## Contoso Inc. Power Platform architecture

After a series of architecture workshops, Contoso Inc. has decided to start designing a high-level architecture for their future Power Platform ecosystem. They decided on their future tenant structure, the environment composition for the planned implementation project, the deployment approach of software components for their clients, and an approach for handling the provisioning of users, licenses, and permissions for various user groups.

Contoso Inc. analyzed the business and IT security requirements and decided to implement strong governance over the planned Power Platform ecosystem. They did this by preparing and implementing user management, application management, and data protection policies supported by Microsoft's Center of Excellence Starter Kit. They decided to install the starter kit on a dedicated Power Platform environment as one of the first steps in the upcoming implementation project.

In this section, we will describe their architectural decisions in more detail.

## Tenant structure

Contoso Inc. is using Microsoft 365 products in their current single-tenant configuration with cloud identities only. For the purpose of implementing the Power Platform ecosystem, it was decided to implement an integration between their own on-premises Active Directory and Azure Active Directory. In order to test all the integration and security features and have a permanent environment for further extensibility, they decided to purchase an additional Azure Active Directory tenant so that the final structure will consist of two tenants, as shown in the following diagram:

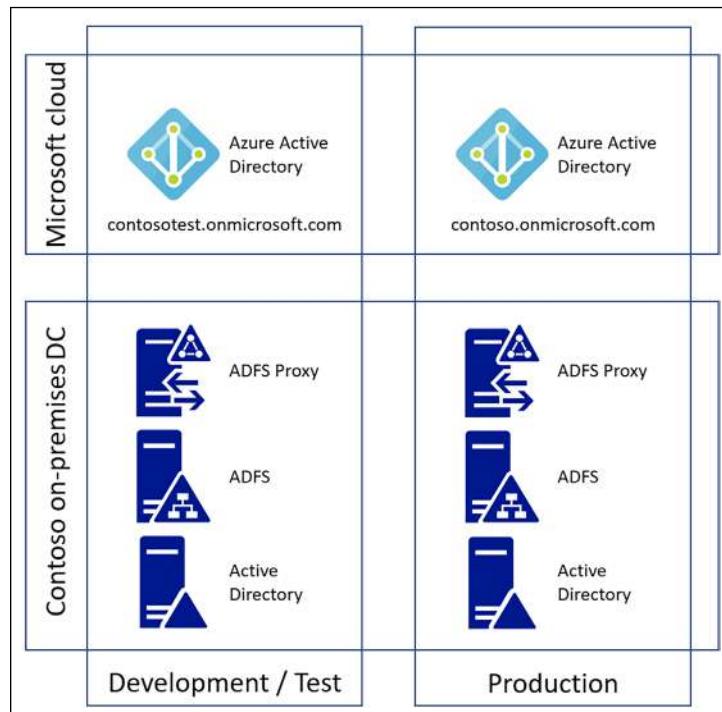


Figure 3.16: Contoso Inc. tenant composition

As documented in the tenant structure, there will be a new development and test tenant in addition to their production tenant. This will be used to develop an integration between their on-premises Active Directory and the cloud AAD. The integration will undergo complex security testing to eliminate any possibility of malicious attacks. The tested and accepted integration configuration will be transferred into the production tenant, which is where all of the Power Platform environments will be deployed.

Contoso Inc. has decided that further details about the specifics of the Active Directory integration will be elaborated after a detailed security analysis.

## Power Platform environments

Contoso Inc. decided to implement a single-tenant strategy for the Power Platform ecosystem so that all Power Platform environments will be always created in the production tenant called `contoso.onmicrosoft.com`.

It was further decided that there will be a single Power Platform **production environment** for worldwide use that contains all the required Dynamics 365 and custom model-driven applications. The environment will be provisioned in the North American Power Platform region.

Since the solution implementation is expected to be complex and long and multiple development streams and releases will be used, Contoso Inc. decided to use the following environment setup:

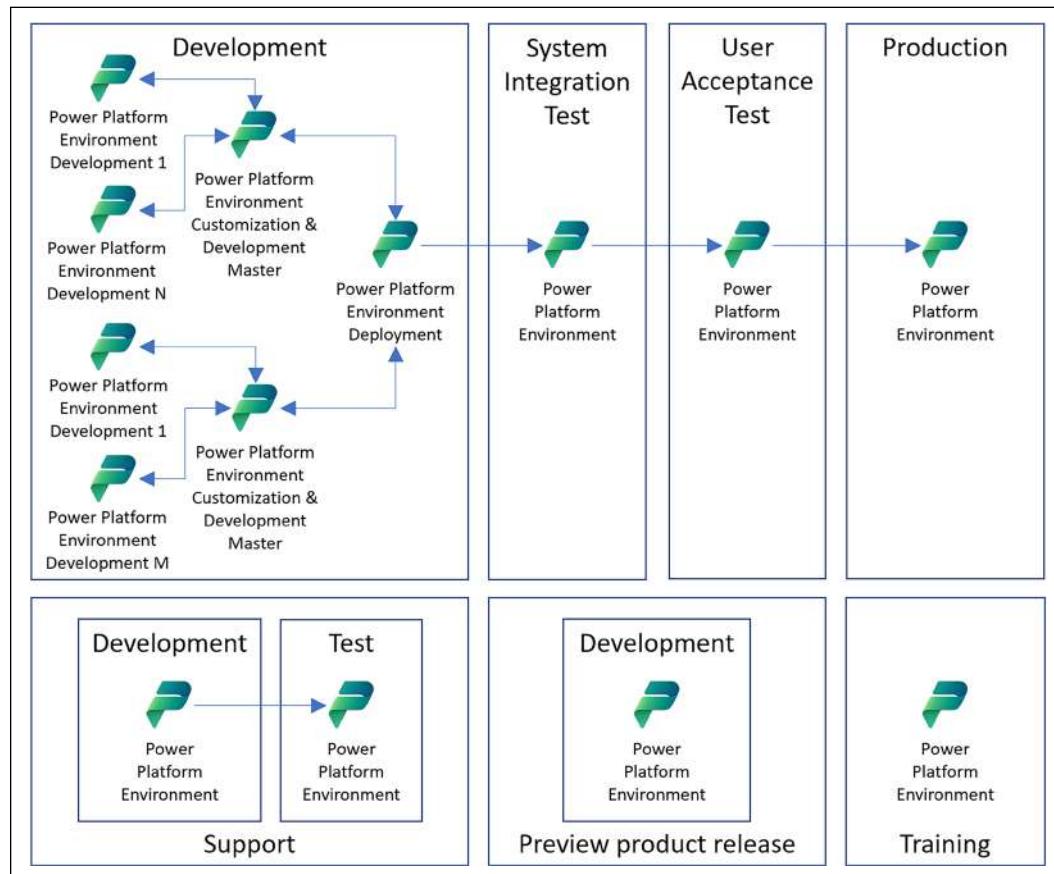


Figure 3.17: Contoso Inc. environment configuration

These environments will be used for the following purposes:

- **Development (1, 2, 3, ... X):** These will be separate environments for individual developers, customizing separate solution subsets, and developing custom-developed artifacts.
- **Customization and Development Master:** This will be used as a consolidation environment to consolidate the solution components from the respective stream.
- **Deployment:** This will be used as the main consolidation environment for merging the streams, packaging solutions, and preparing for exports to the subsequent testing environments.
- **System Integration Test:** This environment will be used for testing the overall solution, along with all the integrations with existing and new IT systems.
- **User Acceptance Test:** This environment will be used for user acceptance tests.
- **Production:** This environment will be the global production environment for all Contoso Inc. users worldwide.
- **Support:** This will be the parallel environment for supporting the current production release of the Power Platform solution, while the next release is being developed on the primary development environments.
- **Preview product release:** This will be an environment where the product preview will always be activated to test the solution's compatibility with the new Power Platform product version.
- **Training:** This will be a separate training environment for end user training.

## **Power Platform clients**

In order to stay compliant with the existing IT policies, Contoso Inc. decided to do the following:

- Use the existing **Microsoft System Center Configuration Manager** to deploy all desktop components of the planned Power Platform solution.
- Establish **Microsoft Intune**, which they have as part of their existing Microsoft 365 subscription, to fully manage all Power Platform components on mobile devices. This will ensure the full compliance of the mobile devices with the IT security policies and protection of the corporate data that might reside on the devices in the Power Platform mobile application components.

When implementing these decisions, Contoso Inc. will stay compliant with the existing IT policies and ensure the mobile devices in the organization stay under full control, with the possibility to remotely protect valuable business information stored on the devices.

## User groups and licensing

Contoso Inc. decided that, in preparation for the Power Platform solution implementation, they would implement the use of **Microsoft 365 security groups** to manage user provisioning into individual Power Platform environments. Contoso Inc. will further establish three main user groups for identity and access management and licensing:

- **Project core team:** This team will need immediate full administrative access to the development environments and the SIT environment. Later, during the course of the solution's implementation, this team will need to have access to any other development and testing environments, such as support, product preview, and so on. To ensure agility, identity and access management, as well as licensing, will be performed at the beginning manually. The members of this team will obtain Azure Active Directory cloud identities, if they do not have them already, and will be assigned full Dynamics 365 licenses.
- **Project extended team:** This team will need user access to the SIT and UAT environment after the first iterative testing process starts. The members will be managed using automated user provisioning and license assignment as pilot users, while the automations will still be in development. The team's members will be assigned the appropriate licenses for the respective Dynamics 365 apps or Power Apps plans.
- **Power Platform solution users:** This team will need user access to the production and training environment later, when the solution is being implemented. The users will be automatically provisioned when the Active Directory federation has been established and the automations are in production. The team's members will be assigned appropriate licenses for the respective Dynamics 365 apps or Power Apps plans.

Contoso Inc. has learned a lot about the Power Platform architecture and has made big preparations for their planned implementation project.

## Summary

In this chapter, you learned about the *Power Platform architecture*, covering topics such as environments, technology, clients, and administration and monitoring, to be able to design a *high-level architecture* for your own Power Platform-based solution. With these skills, you should be able to decide on which regions you should provision your Power Platform instances and whether you need the *Dataverse database* or not.

Furthermore, you should now understand what the usage scenarios for the various environment components are and how to decide on the client types you need to use for your solution. You should also understand what the various possibilities are when it comes to administering and monitoring a Power Platform ecosystem, as well as how to select the best suitable option for your existing IT landscape. Finally, you should now understand the possible consequences of using multiple *tenants* and how to design proper environment composition for your Power Platform implementation project.

In the next chapter, we will focus on the tools and techniques that are used for developing Power Platform solutions.

# 4

## Power Platform Customization and Development Tools and Techniques

In this chapter, we will focus on *tools* and *techniques* used for the configuration, customization, and custom development of **Power Platform solutions**. Power Platform consists of multiple different products, and developing a complex solution requires knowledge and the use of various tools in order to be able to efficiently configure and customize a Power Platform solution, and also perform custom development tasks if necessary.

In this chapter, we are going to cover the following main topics:

- Introducing tools and techniques
- Presenting configuration and customization tools
- Presenting custom development tools
- Application lifecycle management tools
- Contoso Inc. project team workplace setup

### **Contoso Inc. empowering the project team**

Based on the understanding of the Power Platform architecture, Contoso Inc. was able to make key decisions about the Microsoft cloud tenant structure, Power Platform environments, and client types and establish a foundation for governance, administration, and monitoring.

The next step in preparing the Power Platform implementation project will be to understand the toolset necessary for *managing* the project, as well as *configuring, customizing, developing, testing, and deploying* the solution. Contoso Inc. empowers the project team with all they will need to successfully start the project's execution.

Let's start by explaining the concept of the citizen developer versus the IT pro developer we mentioned in *Chapter 3*, to understand the different toolsets used by these two categories of developers.

## **The citizen developer vs. IT pro developer paradigm**

Before we start introducing the various tools that are used for Power Platform solution development, let's introduce the new paradigm in developing business applications by involving and empowering the **citizen developer**. Empowering citizen developers so that they can develop simple applications can solve small business needs and bring business value quickly as opposed to engaging large and complex IT projects.

Let's start by introducing the two main categories of developers.

### **Introducing the citizen developer**

The concept of the **citizen developer** is generally gaining traction in the business world. In the world of Microsoft software, it has been possible for business users to create complex Excel sheets using *formulas*, enhance Excel or Word files with *macros*, or even create small single-purpose databases with Access. But it is the rise of Power Platform that enables the true potential of the citizen developer. A citizen developer can now create departmental or even enterprise-wide applications by connecting to enterprise IT systems and providing automations across them.

### **The IT pro developer**

The **IT pro developer** is a traditional role in the industry. IT pro developers usually have a formal information technology education and are expected to have a deep understanding of hardware and software systems, algorithms, databases, and programming languages. An IT pro developer typically develops advanced IT solutions using *code, integrated development environments, and application lifecycle management tools*.

## Distinguishing between the citizen developer and the IT pro developer

The distinction between a *citizen developer* and an *IT pro developer* can be sometimes blurry and requires an increased level of governance to avoid the negative consequences described in the previous chapters of this book, such as shadow IT, data protection risks, or governance over the whole cloud landscape.

For the purpose of this book, we will distinguish between a citizen developer and an IT pro developer using the following criteria:

- **Citizen developer:** Configures and customizes simple solutions using *graphical design tools* and *simple formulas*
- **IT pro developer:** Develops complex solutions using *programming languages*, *custom development*, and *ALM tools*

### Important note



IT pro developers can also perform all the tasks that citizen developers can, and in many cases, Power Platform experts are on the same level as IT pro developers. The *citizen developer* approach is an additional option to open the area of building apps to business experts from outside the IT domain.

Now that we've explained the difference between the citizen and IT pro developer, let's focus on the toolsets necessary for customizing and configuring business solutions.

## Presenting configuration and customization tools

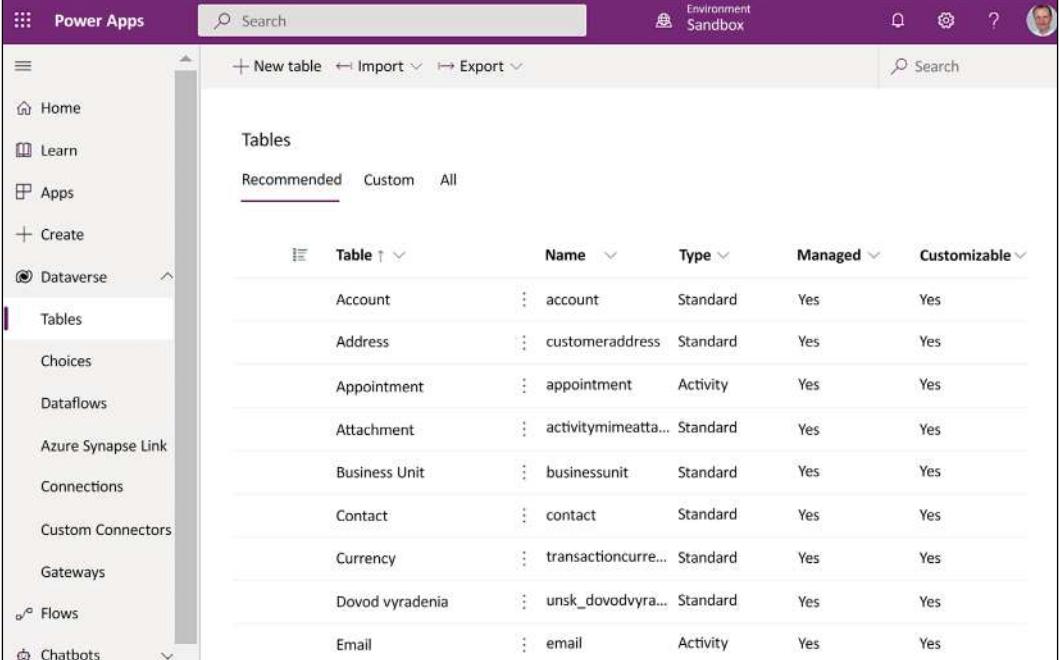
In this section, you will learn about the most important Microsoft and community tools that are used primarily for configuring and customizing Power Platform solutions. This toolset can be used by citizen developers as well as IT pro developers.

### Microsoft Dataverse and model-driven app tools

There are several key configuration and customization tools from Microsoft or third parties. In this section, you will learn about the most important tools that are used for **Dataverse solution development**.

## Power Apps Maker Portal

The main customization tool for building Dataverse and model-driven apps is the **Power Apps Maker Portal**. You can access the portal by going to <https://make.powerapps.com>. This is what you will see when you access this URL:



The screenshot shows the Power Apps Maker Portal interface. On the left, there's a navigation sidebar with options like Home, Learn, Apps, Create, Dataverse, Tables, Choices, Dataflows, Azure Synapse Link, Connections, Custom Connectors, Gateways, Flows, and Chatbots. The 'Tables' option is selected. The main area has a search bar at the top right and tabs for 'New table', 'Import', 'Export'. Below that is a 'Tables' section with tabs for 'Recommended', 'Custom', and 'All'. A table lists various tables with columns for Name, Type, Managed, and Customizable. The listed tables include Account, Address, Appointment, Attachment, Business Unit, Contact, Currency, Dovod vyradenia, and Email.

Table	Name	Type	Managed	Customizable
Account	account	Standard	Yes	Yes
Address	customeraddress	Standard	Yes	Yes
Appointment	appointment	Activity	Yes	Yes
Attachment	activitymymeatta...	Standard	Yes	Yes
Business Unit	businessunit	Standard	Yes	Yes
Contact	contact	Standard	Yes	Yes
Currency	transactioncurre...	Standard	Yes	Yes
Dovod vyradenia	unsk_dovodyra...	Standard	Yes	Yes
Email	email	Activity	Yes	Yes

Figure 4.1: Power Apps Maker Portal

In the Power Apps Maker Portal, the user can do the following:

- Customize the Dataverse database and create or modify tables, columns, relationships, forms, views, charts, and dashboards
- Customize global choices
- Trigger the separate designers for model-driven apps, canvas apps, Power Pages, Power Automate flows, and Power Query dataflows
- Configure the synchronization of Dataverse data in Azure Synapse Analytics
- Manage connections and custom connectors
- Manage on-premises data gateways

- Work with the Power Virtual Agents designer to create and modify bots
- Work with AI Builder to create and modify AI models
- Work with solutions and apps

As you can see, the Power Apps Maker Portal is one of the most important tools for a customizer and a starting point for many other Power Platform configuration and customization tools.

## Model-driven app designer

After the Dataverse database has finally been customized and tailored to the needs of the organization, it is time to create some model-driven apps, which will present a specified subset of Dataverse capabilities to end users. It is recommended to create multiple model-driven apps in order to give every user group just the necessary part of the overall Dataverse solution. This helps avoid confusion and improves user adoption.

Model-driven apps are created using the *app designer*. The designer can be triggered directly from the Power Apps Maker Portal. The designer is illustrated in the following screenshot:

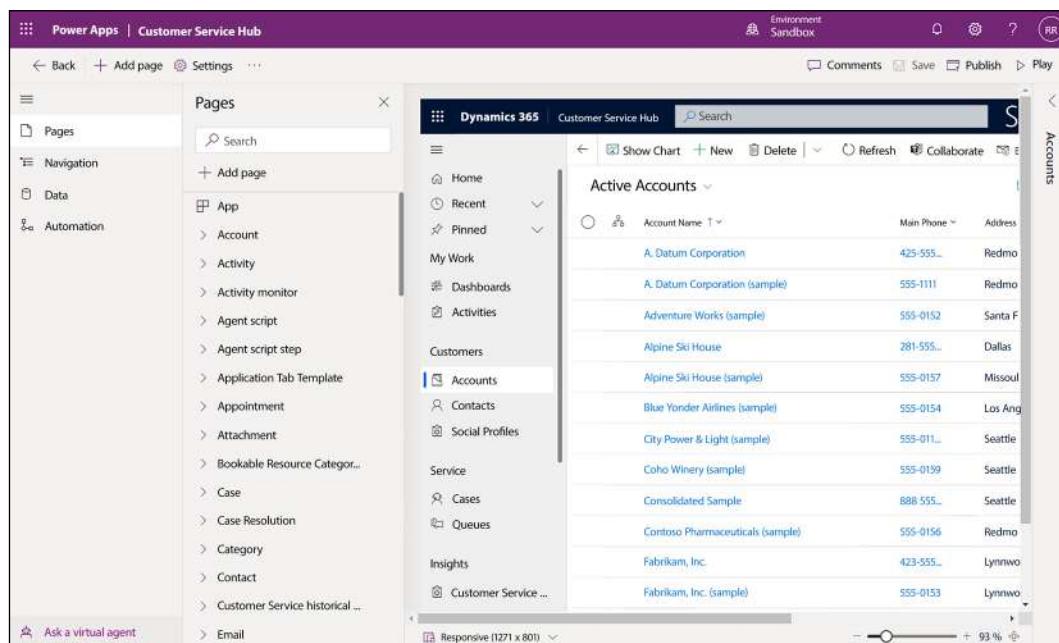


Figure 4.2: Model-driven app designer

The designer provides the following capabilities:

- Adding **pages** to the app: Pages are components, which will become part of the app. Pages can be either Dataverse tables, dashboards, or custom pages, created using the canvas apps technology. For Dataverse tables, it is possible to select which table sub-components will be included – individual forms, views, charts, or table-specific dashboards. This makes it possible to limit the number of sub-components in the app to just those needed by the target user group.
- Extending the **navigation** of the app: The navigation is placed on the left side of the app. The navigation can be designed in a three-level hierarchy described below.
- Adding **Business Process Flows (BPFs)** to the app: The same applies for tables and their sub-components. By selecting a sub-group of all available BPFs, the app will be equipped just with those needed by the target user group.

The navigation (also called the site map) is a three-level hierarchy of navigation within model-driven apps, consisting of:

- **Areas**: Represent the navigation areas that can be selected in model-driven apps in the lower-left corner
- **Groups**: Represent a group of navigation elements within an area
- **Subareas**: Represent the individual navigation elements (table views, dashboards, or other elements)

The model-driven app designer offers a *what you see is what you get* experience – every change in the configuration can be immediately visually verified.

Besides those main configuration and customization tools, there is a multitude of third-party tools that make the life of a Dataverse customizer easier. Some of them are worth mentioning in this chapter.

## Introducing XrmToolBox

XrmToolBox is a community tool. It evolved from a small collection of useful tools for *Dynamics CRM customization* to a platform containing more than 100 tools that had been developed by the author of the toolbox, as well as a broad community of experts. XrmToolBox is a free tool and can be downloaded from the following URL: <https://www.xrmtoolbox.com/>.

Describing all the tools available in the toolbox is not in the scope of this book, but some are very important and worth mentioning. In the area of customizing model-driven navigation, the **Ribbon Workbench** has gained a large amount of popularity and general adoption in the expert community.

## The Ribbon Workbench

The main navigation of model-driven applications is configured using the **Sitemap Designer**. However, the table-specific navigation, the so-called **command bar** (previously called the **ribbon**) that's used to navigate within table forms, views, and sub-grids, can be customized only by exporting the customization XML file, modifying the file, and importing it back into Dataverse. This is cumbersome and error-prone, and the *Ribbon Workbench* provides help with a *graphical user interface* tool that fully replaces the standard capabilities.

### Important note



At the time of writing this book, Microsoft has introduced a true low-code alternative to configuring the command bar. This capability is in preview and does not yet cover everything covered in the traditional command bar configuration. The new capability will provide a configuration-only experience for modifying command bars along with the **Power Fx** expression language as a replacement for JavaScript used today.

Many other tools from this toolbox fill gaps in the Maker Portal by providing capabilities that significantly increase the productivity of a Power Platform customizer and developer.

### Tip



*XrmToolBox* is distributed as a ZIP file, which does not require installation on the user's PC, only unpacking into a folder. This is particularly useful on corporate PCs, which are usually locked down and do not allow individual software installations.

Now, let's focus on other tools dedicated to customizing other Power Platform components. First, we will look into the designer tool for building canvas apps.

## Power Apps Studio

The main customization tool for building canvas apps is **Power Apps Studio**, which can be triggered directly from the Power Apps Maker Portal.

Power Apps Studio is illustrated in the following screenshot:

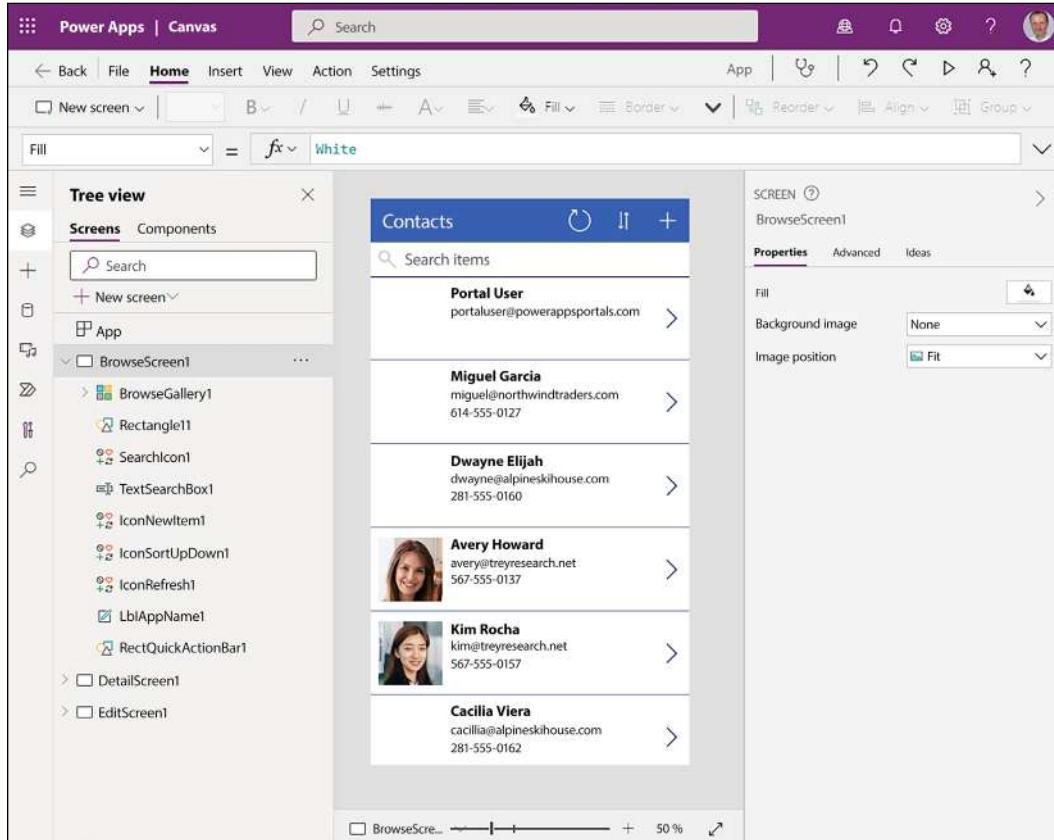


Figure 4.3: Power Apps Studio

Power Apps Studio provides the following capabilities:

- Creating a canvas app from scratch, from a template, or autogenerated one from a data connection
- Designing the user interface of canvas apps (screens, controls, and more)
- Selecting connections for a canvas app using public or custom connectors
- Building business logic using **Power Fx** expressions
- Checking the app with the app checker, testing it, and saving and publishing the app
- Sharing the app with other users, groups, or everyone in the organization
- Triggering advanced tools such as the **Monitor tool** to monitor canvas app activity or **Test Studio** to write and execute application tests

In the following section, we will focus on the designer tool for configuring Power Pages portal applications.

## Power Pages Studio

Power Pages Studio is a graphical designer tool for building web solutions with Power Pages. The tool offers handy customization possibilities for a website's design and configuration, as illustrated in the following screenshot:

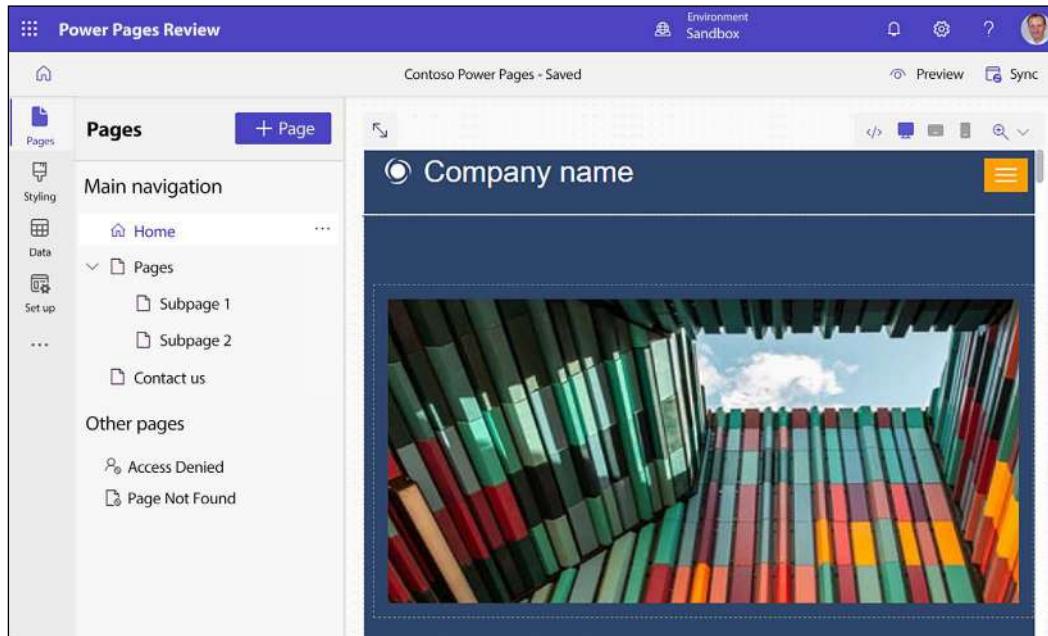


Figure 4.4: Power Pages Studio

Power Pages Studio provides the following capabilities:

- Graphical design of websites – create web pages and place content on them
- Styling of websites
- Configuring access to Microsoft Dataverse data
- Configuration of authentication providers for a web site



**Important note**

Power Pages is a new technology and, as of writing this book, still in preview. It is intended to be a successor to the Power Apps portals technology.

The next tool we are going to present is the designer tool dedicated to Power Automate.

## Power Automate Maker Portal

The main customization tool for building Power Automate flows is the **Power Automate Maker Portal**, which can be triggered directly from the Power Apps Maker Portal or by going to the following URL: <https://make.powerautomate.com>.

The Power Automate Maker Portal is illustrated in the following screenshot:

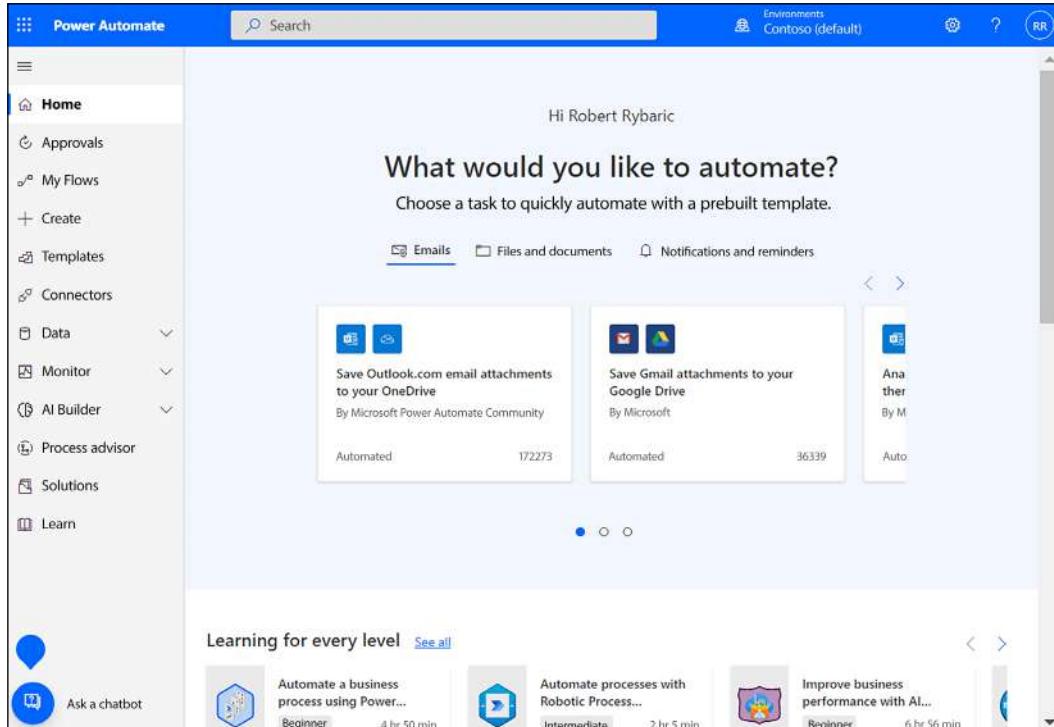


Figure 4.5: Power Automate designer

The portal provides the following capabilities:

- Creating a flow from scratch or using a flow template
- Designing the flow with the business logic components and connectors
- Checking the flow with the flow checker, testing it, and saving the flow
- Sharing the flow with other users or groups

The Power Automate portal also provides an end user workplace experience for using flows, triggering button flows, and managing approval flows. It is also a starting point for creating Power Automate desktop flows and starting the Process Advisor.

Now, we will turn our attention to the chatbot designer tool called Power Virtual Agents designer.

## Power Virtual Agents designer

**Power Virtual Agents (PVA)** designer is a tool that's used to design, test, and publish PVA chatbots. The designer can be triggered directly from the Maker Portal or using the following URL: <https://powerva.microsoft.com/>. With PVA designer, you can do the following:

- Create new bots or delete existing ones.
- Create topics containing trigger phrases, which start the actual conversation with the bot regarding a specific area of interest.
- Create the dialog flow for the bot in a graphical designer environment. The flow of the dialog can contain questions, messages, branching, and other logic.
- Create entities representing certain typical information units, which help the PVA engine understand the intent of a question.
- Call actions from the dialog flow using *Power Automate*. The actions can start secondary processes or retrieve data that can then be used in the conversation flow.
- Finalize a conversation flow with a simple satisfaction survey dialog or transfer the conversation to a human customer support agent.
- Test the bot's functionality.

After a PVA bot has been finalized and works as expected, it is possible to publish it and integrate it with several technologies so that it serves the purpose of automated first-level support. PVA bots can be integrated with, for example, websites, Microsoft Teams, Facebook, and mobile applications. Part of PVA designer is also built-in analytics so that you can analyze the usage and performance of your bots.

## AI Builder

**AI Builder** is a tool used to design AI models and integrate them into other parts of a Power Platform solution. AI Builder can be found directly in the Maker Portal. With AI Builder, you can do the following:

- Create an AI solution based on some of the existing custom or prebuilt AI models.
- Provide data and test the custom AI model.

- Publish the model so that it can be integrated with other parts of Power Platform.
- The finalized AI Builder solution can be used within Dataverse solutions, integrated with canvas apps, or used in Power Automate flows.

Currently, the following **custom AI** models are available:

- Prediction
- Category classification
- Image classification
- Entity extraction
- Object detection
- Document processing

Another group is **prebuilt AI** models, which consists of the following:

- Business card reader
- Identity document reader
- Invoice processing
- Receipt processing
- Text recognition
- Category classification
- Entity extraction
- Key phrase extraction
- Language detection
- Text translation
- Sentiment analysis

The difference between these groups is that while the custom AI models must be supplied with data and trained before they can be used, the prebuilt AI models are ready to use.

## Dataflows designer

The **dataflows designer**, which must be triggered from the Power Apps Maker Portal, is used to create data import jobs using the *Power Query* querying language. The designer has the following capabilities:

- Selecting a data source from a broad range of possible data sources covering all major database systems, as well as many other popular technologies.
- Transforming and filtering the source data using various transformation and filtering options.
- You can loading the data into an existing or new Dataverse table, as well as performing an appropriate column mapping for the data load.
- Load can be a one-time import or scheduled for regular data imports.

The dataflow projects that are created in the designer can be used for data migration or initial data load purposes, or as a permanent solution so that you can integrate certain IT solutions with Power Platform.

## Power BI designer tools

In this section, you will learn about the specifics of designing Power BI solutions. Since Power BI is, technologically, quite different from the other Power Platform components, the designer tools are also different and independent of the Power Platform tools introduced in the previous sections.

Power BI provides three different designer tools for designing standard Power BI reports, paginated reports, and publishing and managing them in the Power BI environment.

## Power BI Desktop

**Power BI Desktop** is the main designer tool for creating Power BI applications. In contrast to the other Power Platform customization tools, Power BI Desktop is a desktop application that needs to be installed locally on the PC of the user. Power BI Desktop can be seen in the following screenshot:

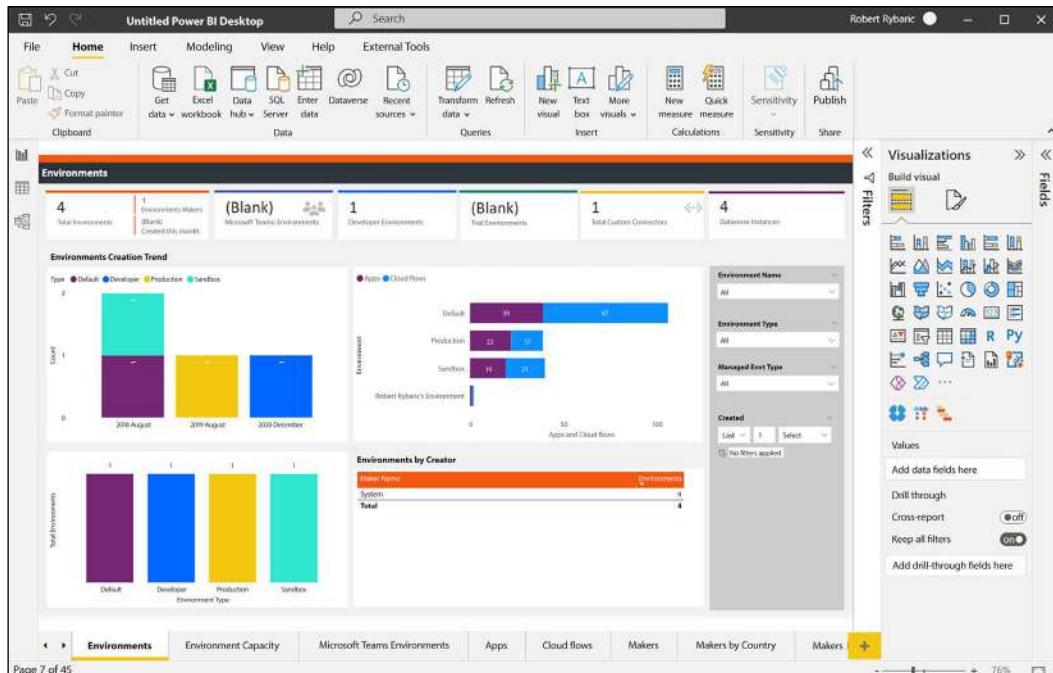


Figure 4.6: Power BI Desktop

Power BI Desktop has the following capabilities:

- Connecting to multiple different data sources, transform and importing the data, and creating a data model
- Creating the required visualizations on top of the data model
- Creating reports
- Publishing reports
- Configuring the RLS security roles

The solutions that are created with Power BI Desktop can be saved locally as Power BI project files with the .pbix extension.

## Power BI Builder

**Power BI Builder** is a designer tool that's used for building **paginated reports**. Paginated reports are typically many pages long and are very much like the traditional **SQL Server Reporting Services** reports, which are used for printing on paper. Power BI Builder is also a desktop application that needs to be installed locally. It can be used to build reports based on the following data source technologies:

- **Azure Analysis Services**
- **Azure SQL Data Warehouse**
- **Azure SQL Database**
- **Microsoft SQL Server (on-premises)**
- **Microsoft SQL Server Analysis Services (on-premises)**
- **Oracle Database**
- **Teradata Database**

Power BI Builder has the following capabilities:

- Creating paginated reports using the aforementioned *data source technologies*
- Publishing paginated reports to **Power BI** (a Power BI Premium license is necessary)

The solutions that are created with Power BI Builder can be saved locally as Power BI project files with the **.rdl** extension.

## Power BI service

The **Power BI service** is the cloud service where Power BI content is deployed and hosted. The Power BI service offers a portal at <https://app.powerbi.com>.

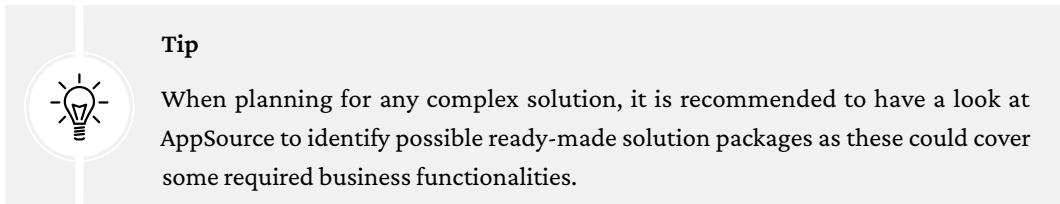
This portal is used to consume Power BI's content, but it also provides certain design and configuration capabilities that need to be performed for newly published Power BI content and that are **not available** in Power BI Desktop. The following services are provided by Power BI:

- Creating dashboards, apps, and workspaces
- Creating dataflows
- Sharing Power BI components with other users

The Power BI service is used by Power BI content creators, but also by end users to access and consume Power BI content. It is possible to find and access a lot of preconfigured Power BI solutions that demonstrate Power BI's capabilities.

## Microsoft AppSource

**Microsoft AppSource** (<https://appsource.microsoft.com/>) is not a configuration, customization, or custom development tool, but rather a public Microsoft solutions repository. You can find hundreds of solutions for Power Platform that come from various ISV partners or directly from Microsoft in this repository.



### Tip

When planning for any complex solution, it is recommended to have a look at AppSource to identify possible ready-made solution packages as these could cover some required business functionalities.

## ISV Studio

**ISV Studio** (<https://isvstudio.powerapps.com/>) is a specific tool dedicated only to Microsoft independent software vendor (ISV) partners, which create and publish applications in Microsoft AppSource. The tool provides mainly analytical and reporting capabilities in order to give the user detailed insight into the behavior of the app within AppSource, including the number of installs, geographical distribution, and version history.

In this section, you learned about the various citizen developer tools that are used for configuring and customizing Power Platform solution components. Depending on your planned Power Platform-based solution, you will need to use some or many of those tools on a regular basis. With the knowledge you've gained from this section, you should have a good understanding of the various tools and their main capabilities.

## Presenting custom development tools

In this section, you will learn about the most important Microsoft and community tools that are used for custom development within Power Platform solutions. This toolset is better suited for IT pro developers.

## Visual Studio

Visual Studio is the traditional major development tool from Microsoft and is used for all custom development work across all Microsoft and many non-Microsoft technologies. Visual Studio is an **integrated development environment (IDE)** that supports not just writing code, but also compiling, building, debugging, and packaging it into solution packages. Visual Studio is available for Windows and Mac PCs. Visual Studio offers several versions, of which *Visual Studio Community* is free of charge.

For Power Platform solutions, Visual Studio is used for creating many different custom development artifacts – preferably those that are heavily dependent on the *Microsoft .NET Framework*, such as the following:

- Dataverse plug-ins
- Dataverse custom workflow activities
- Listeners for Azure Service Bus integration scenarios
- Any Dataverse frontend web resources (HTML, CSS, XML, XSD, JavaScript, and RESX)
- Any custom development for Power BI
- Any Microsoft Azure components used to integrate with Power Platform

To support application lifecycle management scenarios and team development, Visual Studio can be integrated with tools such as *Azure DevOps*.

## Visual Studio Code

Visual Studio Code is a lightweight development tool from Microsoft used preferably for building web and cloud applications. It does not have all the capabilities of its bigger brother. Instead, it requires substantial use of the *command-line interface*, but it also supports some capabilities not available in Visual Studio, such as full *Node.js* support and support for more programming languages. Visual Studio Code is free of charge and available for Windows, Mac, and Linux-based PCs.

For Power Platform solutions, Visual Studio Code is used for creating custom development artifacts that have special development requirements, such as the use of the **Power Apps Command-Line Interface (CLI)**. The **Power Apps Component Framework (PCF)** control is a typical example of such an artifact.

Optionally, any other component that does not depend on the *Microsoft .NET Framework*, including Dataverse frontend web resources and some Azure solutions, can be developed with Visual Studio Code.

## Power Apps Command-Line Interface (CLI)

As mentioned in the preceding section, the **Power Apps CLI** is currently used for developing code components, such as PCF controls. The Power Apps CLI is considered a temporary solution until the code components' development is fully integrated into the standard development and ALM tools. The Power Apps CLI needs to be installed separately on the developer's PC.

## Power Platform extensions for Visual Studio

On Visual Studio Marketplace, there are several free as well as commercial extensions for Power Platform development. Most of these extensions support the development of Dataverse and Dynamics 365 code artifacts – plug-ins, custom workflow activities, and web resources – by providing ready-made solutions and code templates.

For Visual Studio Code, there are also extensions that support Power Platform development. These can be installed directly from the Visual Studio Code environment.

### Tip



Investigate Visual Studio Marketplace or the Visual Studio Code extensions list to find the most suitable modules for increasing developer productivity. Most of these modules are free of charge.

Using extensions is a recommended way to increase developer productivity and prevent unneeded repetitive work.

## NuGet developer tools and assemblies

Microsoft provides a collection of tools and assemblies for custom development, solution packaging and deployment, and migration of configuration data within the **NuGet Developer Tools and Assemblies for Power Apps**.

This package can be installed either from within a Visual Studio project using the *NuGet Package Manager* or individually downloaded from the NuGet repository.

While the assemblies will be discussed later in this book when we discuss Power Platform extensibility, the package contains five tools that are important for any Power Platform developer, of which two are used in the area of custom development. We are going to describe the capabilities of these tools next.

### The code generation tool

This tool is used to generate classes for early-bound programming against the **Organization Service (SOAP)** API used by Dataverse. The early-bound style has a lot of benefits over the late-bound style by providing IntelliSense within the development environments, as well as avoiding typo-based code bugs in the variables and methods related to Dataverse metadata. These programming styles will be discussed in detail in *Chapter 8, Microsoft Power Platform Extensibility*.

## The plug-in registration tool

This is the main developer tool for registering several different development artifacts within the Dataverse solution, including plug-ins and custom workflow activities, as well as several cloud connections such as Azure Service Bus, Azure Event Hubs, and webhooks. The registered artifacts or connections serve as event handlers that propagate Dataverse data to the recipients.

## XrmToolBox

XrmToolBox, which we described earlier in this chapter, also contains many very useful tools for Power Platform custom development. It provides tools for enhanced plug-in management, enhanced early-bound class generation, *FetchXML* building, enhanced *solution management*, browsing *Dataverse metadata*, enhanced support for building *PCF controls*, enhanced *Power Apps portals* management, and enhanced web resource management.

## Postman

Postman is a third-party tool used for developing and testing applications that communicate with a *Web API interface*. The tool offers a paid as well as a free version and is available from the vendor's website: <https://www.postman.com/>.

The tool has the following capabilities:

- Creating development and testing environments and saving configurations.
- Supporting a broad range of authentication scenarios, ranging from no authentication, basic authentication, and *OAuth* to certain vendor-specific authentication types.
- Easily building requests, executing them, and analyzing the responses.

The tool is useful for developing applications against the *Power Platform Dataverse Web API* to quickly and effectively test every type of authenticated request against the API.

## CRMRestBuilder

CRMRestBuilder is a community tool for graphically building various **REST-based request strings** to be used in code when communicating with the Power Platform Dataverse API. The tool is free of charge and is available from the following website: <https://github.com/acornsoft/CRMRESTBuilder>.

This tool is very useful since it can save a lot of detailed coding work when it comes to building *REST* request code fragments.

## Testing tools

An imperative part of every software development project is *testing*. In this section, you will learn about suitable tools for testing various parts of Power Platform solutions.

### Testing the user interface

**Easy Repro** is a testing tool that supports automated **user interface (UI)** tests for model-driven applications. Microsoft provides the Easy Repro testing solution as a free-of-charge Visual Studio solution package. This can be downloaded from the following GitHub repository: <https://github.com/Microsoft/EasyRepro>.

With Easy Repro, it is easy to create test scenarios for almost every part of a model-driven application. The tool supports testing navigation, views, forms, charts, dashboards, reports, workflows, and much more with a predefined set of test packages.

### Testing backend components

Testing backend components for Dataverse applications (plug-ins, custom workflow activities, and so on) is not straightforward, since these components are deployed in the Dataverse database and run in the cloud, where a developer cannot establish a connection and debug the components. There are various approaches to testing these components, such as using *tracing* or using the *plug-in profiler*. However, you can use an alternative approach and test locally using mocked or faked data by leveraging some of the supporting frameworks. Popular frameworks for this type of testing are as follows:

- **FakeXrmEasy** (<https://dynamicsvalue.com/>)
- **FakeItEasy** (<https://fakeiteeasy.github.io/>)

Both are free-of-charge community tools without any official support.

### Using network traffic analyzers

For some debugging and testing scenarios, it is very important to be able to analyze the detailed network traffic between the Power Platform application and the local device. There are several **network traffic analyzers** on the market that can be successfully used for this purpose. Among the most widely used analyzers in the Power Platform community and with similar capabilities are the following:

- The free-of-charge *Fiddler* (<https://www.telerik.com/fiddler>)
- The commercial *HttpWatch* (<https://www.httpwatch.com/>)

Both have similar capabilities to capture and analyze the details of the network traffic between your local PC and the Power Platform solution in the cloud.

In this section, you have learned about the various tools used for custom development, if required as part of building a Power Platform solution. While simple solutions might not need any custom development efforts, when building complex solutions with advanced requirements for business functionalities and the user interface, or with complex integration and data migration requirements, custom development might be necessary. Now, you should have a good understanding of these tools and be able to use them in your daily work.

## Presenting application lifecycle management tools

For large software development projects, it is necessary to use application lifecycle management tools to manage tasks, including collaboration, development, and builds, as well as to control source code versioning. In this section, you will learn about the tools that can be used for the **application lifecycle management (ALM)** of complex Power Platform solutions.

### NuGet developer tools and assemblies

NuGet developer tools and assemblies contain three tools that can be used for managing application lifecycle processes when developing a Power Platform based solution – specifically when using Power Platform solution deployment. In the following section, you will learn about the capabilities of these three tools.

#### The configuration migration tool

This tool is used to create a **data migration package** to migrate smaller data amounts from one Dataverse environment into another. It is used preferably for migrating static configuration data from several tables in one package. The advantage of this tool is that it can combine data from many tables into a single package while preserving relationships between records in the tables.

#### The package deployer tool

This tool is used to create **deployment packages** to deploy complex solutions from one Power Platform environment into another. The tool can deploy, in one single deployment process, multiple Power Platform solution packages, along with a data migration package that's been created with the *configuration migration tool*. In addition, the tool can perform post-deployment tasks that are executed after the whole deployment is successfully finished.

## The solution packager tool

This tool is used to extract and repack the components from a Power Platform solution package for the purpose of managing them in a source control system.

## Azure DevOps

The main ALM tool for any large Power Platform solution development should be **Azure DevOps**. Azure DevOps is a software development tool used for collaborative work planning, code development, and building and deploying applications.

This tool can be seen in the following screenshot:

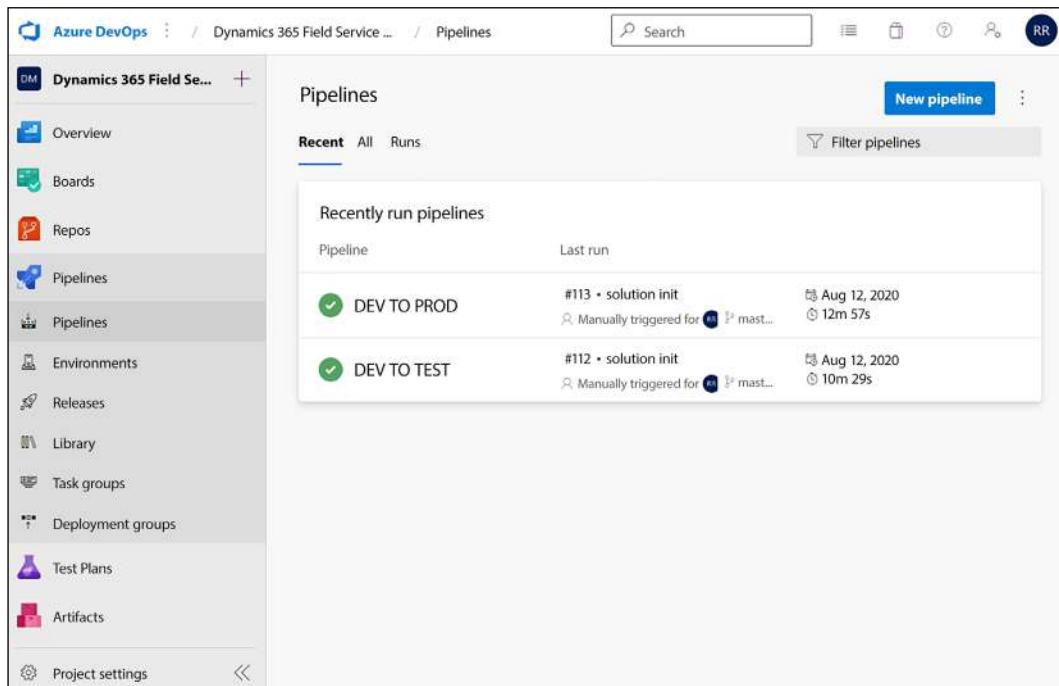


Figure 4.7: Azure DevOps

Azure DevOps provides the following main capabilities:

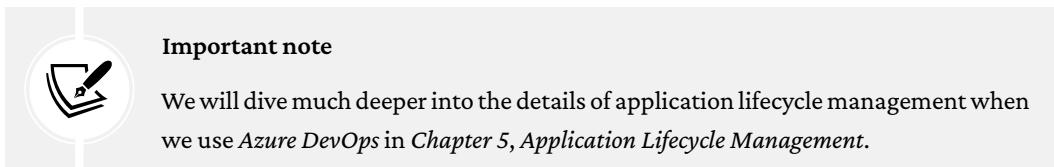
- Agile planning and work tracking tools in **Boards**
- Repositories for source control in **Repos**
- Build and release automation tools in **Pipelines**
- Testing support with **Test Plans**

- **Artifacts** for storing various tools and packages used in the build and integration processes

These capabilities are used in different phases and for different purposes in a project, such as for agile project management, solution development and source control, and solution deployment and testing.

Specifically, for the purpose of Power Platform ALM, Microsoft provides a set of build tools called **Power Platform Build Tools** with the following capabilities, all of which can be integrated into the DevOps pipeline:

- **Environment tasks** to create, delete, back up, and copy environments
- **Solution tasks** to export, import, pack, and unpack solutions, set solution versions, deploy packages, and publish customizations
- **Quality check tasks** to perform quality checks using the Power Platform Checker
- **Helper tasks** to support the installation of the Power Platform tools package at the beginning of the pipeline execution



#### Important note

We will dive much deeper into the details of application lifecycle management when we use Azure DevOps in *Chapter 5, Application Lifecycle Management*.

## Contoso Inc. project team workplace setup

The Contoso Inc. project management team has made itself familiar with the tools available for Power Platform solution development and has made some preliminary decisions.

### Enabling the core project team

The core project team will be equipped with proper licenses and access rights to use the following:

- Power Apps/Power Automate Maker Portal, along with all designers and studios
- Power BI service
- Power Virtual Agents designer

Contoso Inc. will provide the necessary licenses and configure their **Microsoft Endpoint Configuration Manager** so that it can deploy the following tools onto the workstations of all the core team members of the project:

- Power BI Desktop

- Power BI Report Builder
- Visual Studio
- Visual Studio Code
- NuGet Developer Tools and Assemblies
- XrmToolBox
- CRM Rest Builder
- Easy Repro
- Fiddler

Contoso Inc. will establish an Azure DevOps instance and provide access to all the core project team members. **Power Platform Build Tools** will be installed on the Azure DevOps instance.

## Enabling citizen developers

Contoso Inc. decided not to provide access for citizen developers to the Power Platform tools in the first place, instead establishing a full level of governance supported by Microsoft's *Center of Excellence Starter Kit* first. The empowerment of citizen developers will be established gradually as the development of the Power Platform solution progresses and management gets a better idea of what areas of their Power Platform solution could be comfortably handled by citizen developers.

## Summary

In this chapter, you learned about the most important tools that are used for developing a Power Platform solution. With this knowledge, you should be able to prepare a perfect working environment for your Power Platform ecosystem in which citizen as well as IT pro developers are empowered to build their first Power Platform business solutions.

Now that we have complemented our knowledge of Microsoft cloud services and architecture with a detailed view of what toolset can be used to build Power Platform solutions, we can turn our attention to the final topic concerning the design of Power Platform solutions, **application lifecycle management**.

# 5

## Application Lifecycle Management

In this chapter, we will analyze the possibilities of using the **Application Lifecycle Management (ALM)** approach in Power Platform solution implementations. Power Platform solutions are heterogeneous and complex, and a proper ALM approach is key for a well-structured and organized implementation project execution.

In this chapter, we will cover the following main topics:

- Overview of application lifecycle management
- Power Platform solution management
- Azure DevOps for the Power Platform, Power BI, and other components
- Application lifecycle management best practices
- Contoso Inc. ALM strategy

### Contoso Inc. implementing application lifecycle management

In the previous chapter, Contoso Inc. was able to define the workplace setup for the members of the future project team. Now, Contoso Inc. needs to start analyzing the technical possibilities of the Power Platform for managing complex **solution developments** from the governance point of view. Large IT projects in general need deep governance to manage processes properly, keep defined structures, save all project artifacts, automate deployment procedures, etc. Power Platform offers a lot of support to achieve a good level of governance. In this chapter we will focus on the most important governance features, tools, and best practices, especially the Power Platform solution management and Azure DevOps.

## Understanding application lifecycle management

**Application lifecycle management (ALM)** is a crucial governance principle and approach for developing every software solution. Power Platform solutions can be everything from very simple for a few users in one country to very complex, international, and serving thousands of users across the globe. While the simplest solutions might not necessarily need a highly structured *ALM approach*, for larger solutions, proper ALM is inevitable. In this section, you will get a basic understanding of the role of ALM in Power Platform solutions and the factors influencing the project complexity.

### Environment complexity

Power Platform project complexity is influenced by a number of factors, which could dictate the use of a proper *ALM strategy* to be successful. The most important are as follows:

- **Requirements complexity:** The complexity might, for example, require heavily using custom development, which will influence the deployment complexity. If you need to deploy not just the pure Power Platform solution, but a number of cloud and/or on-premises custom development artifacts as well, the deployment procedures might be anything but trivial.
- **Geographical coverage (local versus global):** The geographical coverage might require performance optimization as one of the key solution requirements to achieve a very good end user experience regardless of where on the globe the users are located.
- **The number of users and their geographical distribution:** The number of users might require user provisioning and maintenance automations and regional distribution of administration capacities.
- **The number of supported languages:** The number of languages might influence solution translating requirements and documentation into several languages.
- **Existing IT landscape and legacy systems:** The existing IT landscape might require the use of specific customizations, compatible with the IT landscape.
- **Security requirements:** Specific security requirements might increase the efforts necessary for designing the authentication and authorization within the Power Platform.
- **Integration requirements:** The number and complexity of existing IT solutions to be integrated with the Power Platform might require the use of various additional technologies and might significantly increase development and deployment efforts.

- **Data migration requirements:** The data migration requirements can heavily impact the design and development of the solution as well as the deployment schedule and complexity.
- **Change management requirements:** The level of maturity of the organization and future users of the solution might require additional efforts for training and documentation.
- **Operational and maintenance requirements:** The ability and experience of the organization to operate and maintain a complex cloud solution can influence the project duration and costs.
- **Human factors (delivery teams, customer teams, and use of offshore capacity):** The human factors might significantly influence the project delivery efficiency and development and deployment complexity.

As you can see, there are a lot of environment complexity influences that might require organizations to consider advanced ALM strategies to make the project manageable. In the next section, we will discuss the Power Platform solution complexity factors, suggesting the need for proper ALM strategies.

## Power Platform solution complexity

A very important factor to consider is the nature of the Power Platform as a very heterogeneous and complex ecosystem. Unlike a pure custom development solution, a Power Platform solution, especially a complex one, consists of various components. A complex Power Platform solution can consist of the following types of components:

- Microsoft Dataverse customization, such as the extension of the data model, the use of various automations, and many more
- Microsoft Dataverse backend custom development, such as plug-ins or custom workflow activities
- Microsoft Dataverse frontend custom development, such as JavaScript event handlers, PCF controls, and many more
- Microsoft Dataverse configuration data
- Model-driven apps
- Canvas apps and canvas app components
- Power Automate flows
- Custom connectors for canvas apps and Power Automate flows
- Power BI solution artifacts

- AI Builder models
- Power Virtual Agents chatbots
- Power Pages portal solutions and their configurations
- Microsoft Azure solutions
- On-premises custom development artifacts

There are many other components, but not all of them can be listed here. Therefore, for brevity, we are listing the most important ones.

The heterogeneous nature of Power Platform solutions also dictates the need to have several different experts in the project team with various specializations, so the various solution artifacts might need to be developed in a heterogeneous team structure.

## **ALM for the Power Platform**

As described in the previous sections, a Power Platform-based solution can be very complex and it would be almost impossible to manage such a project without the help of proper ALM support. The ALM required for Power Platform-based solutions needs to specifically fulfill the following requirements:

- Provide a container to enclose all of the different artifacts of a Power Platform-based solution to have a common structure, management, and simplified deployment.
- Provide a tool to manage the Power Platform-based solution in terms of source control, versioning, and automated deployment.

The described ALM requirements for Power Platform-based solutions are very well addressed with the following main concepts and tools:

- Power Platform solution management
- Deployment automation tools such as Azure DevOps with the Power Platform Build Tools or GitHub with the Power Platform actions

Both of these concepts will be described in more detail in the next sections of this chapter.

## Introducing solutions management

Solutions management in the Power Platform is a natural evolution of the former *customization file (XML file) approach*, which was used in early versions of Dynamics CRM for transporting a customization from one Dynamics CRM instance into another. Solutions management is a primary deployment feature for Power Platform solutions. A Power Platform solution is a container, containing everything a solution consists of. The purpose of the solution is to deploy the solution in its entirety from one environment to another.

Solutions management requires a *Microsoft Dataverse database* to be created in the Power Platform environment. In environments without Microsoft Dataverse, this capability is not available.

In the next sections, you will learn a lot of details about the Power Platform solutions management capability.

### Overview of solutions

In the Power Platform, the **solution** is the key vehicle for supporting ALM. From the ALM point of view, the solution package is a **container** for distributing solutions between environments.

Technically a solution package is a ZIP file consisting of a structure of folders and files, containing all solution artifacts.

In the past, the solution package contained only the solution components of model-driven or Dynamics 365 applications. With the rise of the Power Platform, the solution package now encompasses all Power Platform solution artifacts. As already mentioned in the previous chapters, Power BI is technically so different from the other Power Platform components that it cannot be currently distributed using the Power Platform solution management. Possible Microsoft 365 or Microsoft Azure components also could not be part of the solution package.

Using solution packages is *mandatory*. There is no other way for transporting solution components between environments.

Currently, the following solution components can be included in a Power Platform solution package, but the development of the Power Platform is very fast and the number of possible components to be included in a solution is growing constantly:

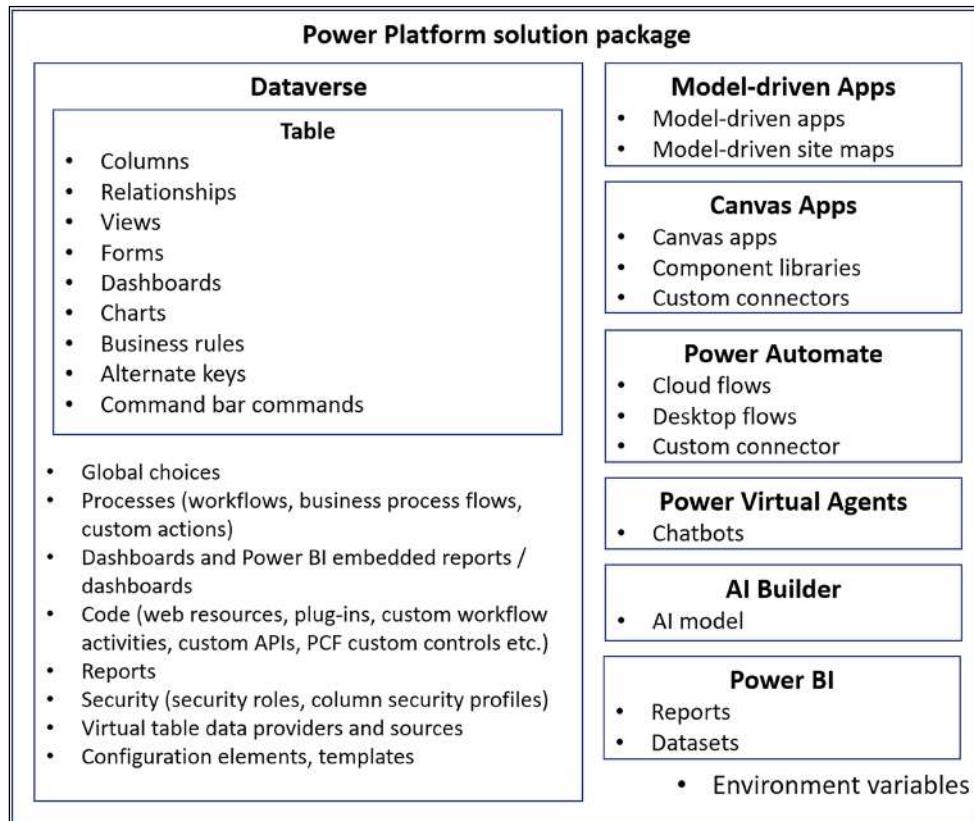


Figure 5.1: Solution package structure

The purpose of some of the respective artifacts will be explained later in this book. From the ALM point of view, the **environment variable** artifact plays a very important role so we will describe it in the following section.

## Environment variables

**Environment variables** are a new concept in the Power Platform and their primary purpose is to support the ALM automations. An environment variable can be created manually in the maker portal or with code. The variable is managed within a specific *Dataverse table* and can be accessed from any Power Platform solution component. The variable can be one of the following data types: **text**, **JSON**, **decimal number**, and **two options (Yes/No)**.

It should be used preferably to store configuration data **specific to environments**. When used properly, the environment variables can completely avoid storing any environment-specific configuration settings anywhere in the solution components. For example, URLs or email addresses are environment specific and need to be addressed within canvas apps or Power Automate flows. Instead, the solution component can reference environment-specific information from the variables, such as the following:

- Environment-specific URLs, email addresses, and connection strings
- Registration values for *ISV* solutions
- Initial setup and configuration parameters used across apps and other settings

The use of environment variables is fully supported in automation tools such as Azure DevOps. In Azure DevOps, it is possible to create a deployment settings file, containing the required current values for the variables so that they are set during the deployment to the target environment.

## Solution properties

Every solution package has some key properties that influence all components inside the solution, as well as the management of the solution package itself. Besides the solution name, there are some additional key properties. Let's take a look at these properties in the subsequent sections.

### Solution publisher

The **solution publisher** is a record in a separate *Dataverse table*, referenced in the solution package. The publisher represents the individual, team, or organization responsible for creating the solution.

Solution publisher reference is mandatory for every solution package. Multiple solution packages can share the same solution publisher. The solution publisher specifies the following key parameters:

- **Prefix:** This is a text string that will be put in front of the technical/schema name of each artifact created within a solution referencing the solution publisher.
- **Choice Prefix:** This is a five-digit autogenerated integer value between 10.000 and 99.999 that will be used as a prefix for all Dataverse columns of the **Choice** datatype, created within a solution referencing the solution publisher.

Besides these two key parameters, it's possible to enter full publisher contact details, which are usually used by ISV companies, distributing Power Platform solutions publicly using *Microsoft AppSource* or other channels.

## Solution version

The **solution version** is a mandatory parameter and is used to manage the patching and updating capabilities of the Power Platform **solutions engine**. It is necessary to respect the recommended structure of the solution version, consisting of four numeric values separated with dots, for example, **1.0.10.125**.

### Important note



The solution version is only relevant to the solution within Microsoft Dataverse. When automating solution deployments, the solution checked into a repository, for example, the Azure DevOps internal Git repo, will have another versioning.

## Configuration pages

The **configuration page** is an optional solution package setting, used preferably by ISV companies to provide an option for entering initial configuration data, license key, and other important details after a solution package is installed on an environment. A configuration page is technically an *HTML web resource* that must implement the configuration setting capability.

In the following section, we will focus on a very important topic: the description of the solution types, and how they are used in deployment scenarios.

## Solution types

One of the key concepts of the Power Platform solution packages is the **solution type**. It is very important to understand the role of the two solution types described below to specify deployment processes correctly and in alignment with Microsoft's recommended best practices.

There are two basic solution types. We will learn more about these solution types and their features in the following sections.

## Unmanaged solution

When a solution is created, whether done manually or with code, it is created as an **unmanaged solution**. Both unmanaged and managed solution types share many identical characteristics, but there are also key differences:

- An unmanaged solution can be created manually or with code.
- An unmanaged solution is considered *open*, where all settings and the contained components can be modified.

- An unmanaged solution can be exported from a Power Platform environment as an unmanaged or as a managed solution.
- If an unmanaged solution is deleted from an environment, the contained components are not deleted; basically, they stay completely unchanged and can be found in the **Default** solution, described below.

## Managed solution

A **managed solution** can only be created in the process of exporting an unmanaged solution. The typical characteristics of a managed solution are as follows:

- A managed solution cannot be created directly.
- A managed solution is considered *closed* where, depending on the configuration, some or almost all settings and contained components are locked down and cannot be modified.
- A managed solution can be imported into an environment but cannot be exported out of an environment.
- If a managed solution is deleted from an environment, all of the contained components are also deleted; the environment will have a structure as if the solution had never been imported.
- Managed solutions support updating, upgrading, and patching.

## Default solution

In every Power Platform environment, an unmanaged solution called **default** is always automatically created and connected to an automatically created **default** publisher. The default solution always contains all solution artifacts stored in the Power Platform environment. The default solution cannot be exported and is not directly used for any purpose.

## Common Data Services Default Solution

Another solution, automatically created in every Power Platform environment, is the **Common Data Services Default Solution** (still using the old name of Microsoft Dataverse). This solution is also connected to a specific automatically created publisher called **CDS Default Publisher**. This solution has no practical use and serves just as an example of how to create real solutions for deployment purposes.

## Managed properties

**Managed properties** are used to specify which settings of a solution artifact can or cannot be modified on the target environment when a solution is imported as **managed**. Managed properties can be configured for several different solution artifacts and vary depending on the artifact type. For most of the artifact types, there is just a simple setting, whether the artifact type *can* or *cannot* be customized. For some artifact types, there are more settings. A few of these artifacts are mentioned in the following sections.

### Forms, views, charts, and dashboards properties

For these artifact types, the following managed properties are available:

- Can be customized
- Can be deleted

### Columns properties

For columns, the following managed properties are available:

- Can be customized
- The display name can be changed
- The requirement level can be changed
- Additional properties can be changed

### Tables properties

Tables have the most managed properties available:

- Can be customized
- The display name can be changed
- Additional properties can be changed
- New forms can be created
- New views can be created
- New charts can be created
- Hierarchical relationships can be changed
- Change tracking can be changed
- Sync to external search index can be changed

A typical use of managed properties is for ISV companies, which usually try to protect their intellectual property by restricting the modification possibilities of their solution packages.

Next, we will deal with solution dependencies and segmentation as another important and complex topic to understand when designing a proper ALM strategy for a Power Platform solution.

## Dependencies and solution segmentation

Within the solutions, there are many dependencies where some of the solution components depend on others. This can be between components, but also on the level of subcomponents of the components, which is specifically the case in Microsoft Dataverse.

To better understand the *dependencies* in the solutions and the approach of **solution segmentation**, it is important to first understand how the solutions are layered in a Power Platform environment.

## Solution layering

In the *Dataverse*, there are three layers for solutions, as illustrated in the following diagram:

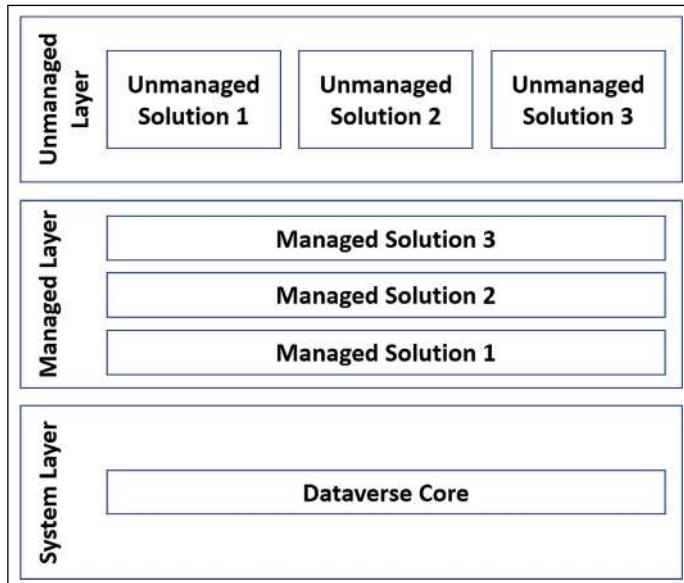


Figure 5.2: Power Platform solution layering

The layers and their behavior are as follows:

- **System layer:** This contains the core *Dataverse tables*, and there is nothing a developer can do to influence this layer. The core tables are protected and cannot be deleted, nor can their components be deleted.

- **Managed layer:** This layer contains all managed solutions imported into an environment. The key concept in the managed layer is the **installation order**. Every managed solution is installed on top of all the already installed managed solutions and may depend on their components. All purchased Microsoft first-party applications (*Dynamics 365 applications*) are also present in the managed layer as managed solution packages.
- **Unmanaged layer:** This contains all unmanaged solutions created or imported into an environment. Unlike managed solutions, there is **no layering** and there are no dependencies between unmanaged solutions. They exist in parallel with each other.

Next, it is important to understand the layering behavior or, in other words, the result of having different customizations in several layers. You will learn about it in the next section.

## Layering behavior

The *solution layering* approach has a specific impact on the result of customizations of the same component in the described layers. The customizations are applied, combined, and merged across all layers, from the bottom up. An individual component, which might be part of the core *Dataverse data model*, could be customized in some managed solutions in the **managed layer** and also in the **unmanaged layer**, as illustrated in the following diagram:

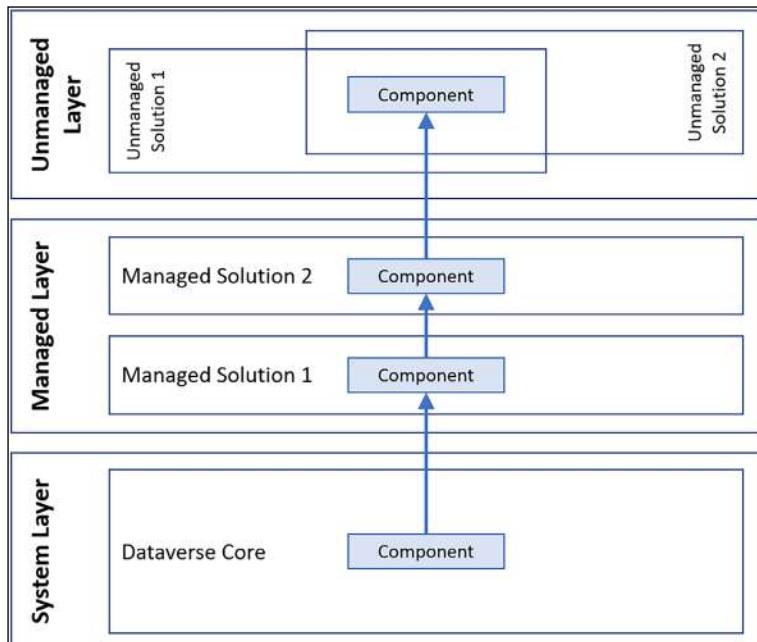


Figure 5.3: Solution layering behavior

As we can see, the ultimate customization of a certain solution artifact can be a combination or merge of several partial customizations bottom-up. This behavior is also one of the sources of solution dependencies.

To better understand layering behavior and the result of customizations of a certain solution component, the *Dynamics 365 legacy customization tool* provides a function called **solution layers**. It presents a list of all solutions containing the particular solution component in the order in which they were installed onto an environment.

## Solution dependencies

Solution dependencies always work bottom-up so that any component of a customization can have dependencies on the solutions below but not above them. This dependency behavior has two main consequences that need to be considered when planning for the whole solution management concept:

- A solution cannot be imported onto a target environment when any dependencies of the solution components are not yet available on the target environment. In the example illustrated in *Figure 5.3*, it would not be possible to import **Managed Solution 2** into an environment without having **Managed Solution 1** already imported.
- A solution cannot be deleted from an environment when any components of any solution preceding the respective solution depend on any components of the respective solution. In the example illustrated in *Figure 5.3*, it would not be possible to delete **Managed Solution 1** prior to deleting **Managed Solution 2**.

To help the developer to identify solution dependencies, the Power Apps maker portal provides a function called **Show dependencies** on the solution level, which presents a list of all identified dependencies. In the *Dynamics 365 legacy customization tool*, there is a possibility to show dependencies even on the level of the individual solution components.

## Solution segmentation

To reduce the negative impact of solution dependencies, especially in complex solution structures with a lot of layered managed solutions, the concept of **solution segmentation** was introduced with *Dynamics 365 version 8.0*. Solution segmentation is used on the level of Dataverse tables and makes it possible to include only a **subset** of all of the subcomponents of a table into a solution package, not just the whole table as it was before. This is especially useful in larger project teams, where a group of project team members works on customization, possibly on the same component, in parallel. Proper use of segmentation can minimize collisions. Another important use case for solution segmentation is the concept of **solution patching and updating**, described in the next section.

## Patching and updating solutions

Together with the concept of solution segmentation, there was also a concept of **patching** and **updating** introduced with *Dynamics 365 version 8.0*. This concept dramatically changes and improves the process of managing and deploying smaller or larger changes of existing solutions, significantly supporting the *ALM approach* within Power Platform solutions.

### Solution patch

**Solution patches** are small changes in some of the solution components, usually used for bug-fixing purposes or minimal changes in functionality. A solution patch has the following specific behavior:

- A solution patch is created from an existing unmanaged solution in the development environment using a functionality called **Clone a patch**.
- Once created, a new solution package is generated and the package's version number must be increased in the last two numeric parts of the solution version number, for example, from version 1.0.10.25 to 1.0.10.26.
- The solution from which the patch was created will be locked and cannot be modified.
- All components that need to be modified must be included in the patch solution and modified there. Patch solutions should be kept small, so it is recommended to use solution segmentation to include only the minimum necessary number of components.
- Once the modifications are finished, the patch solution is exported as managed and imported into the target environment, where it will be placed immediately on top of the original solution from which it was inherited.
- Solution patching can be repeated, so a single solution can have several subsequently created and applied patches.

Solution patching is a very useful concept and helps to keep deployment processes clean and light compared to deploying a full solution package every time for even the smallest solution modifications.

### Solution updates

**Solution updates** are larger changes to some of the solution components usually used to provide some new functionality and, at the same time, consolidate previous patches into a new solution. A solution update has the following specific behavior:

- A solution update is created from a (previously patched) unmanaged solution in the development environment using a functionality called **Clone solution**.
- Once created, a new solution package is generated and the solution's version number must be increased in the first two numeric parts of the solution version number, for example, from version 1.0.10.26 to 1.1.10.26.
- The new, updated solution will automatically consolidate all existing patches of the original solution and will be ready to perform all required modifications.
- Once the modifications are finished, the updated solution is exported as managed and imported into the target environment. In the target environment, the updated solution will replace the original solution together with all possible patches of the original solution.

There is one important aspect of solution management and that is **removing unwanted components** from the target environments, in cases where some components from a previous solution version are no longer needed and we would like to remove them. Since the solution management engine is generally additive, removing unwanted components was previously a cumbersome multi-step process using a *transient solution*. With the concept of **solution updates**, the component removal complexity is resolved, as illustrated in the following screenshot from the solution import process:

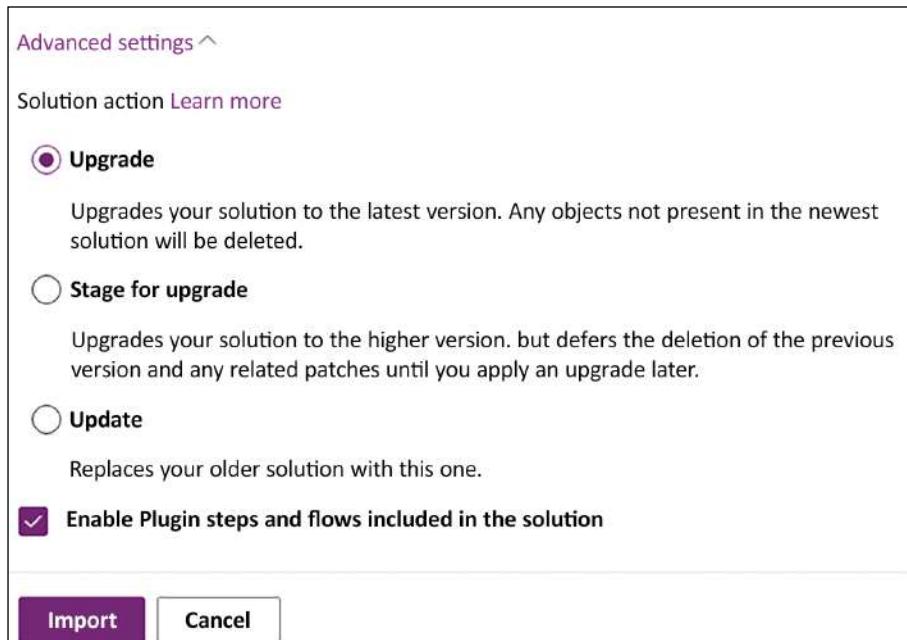


Figure 5.4: Solution update options

Selecting one of the first two **solution actions** (**Upgrade** or **Stage for upgrade**) will remove the unwanted components (components that were removed from the updated solution) from the target environment.

The second option, **Stage for upgrade**, provides the additional possibility to temporarily keep both the original and the updated solution in the environment for possible data migration or any other housekeeping actions before the original solution is finally deleted in the subsequent step. The second step is initiated by the **Apply the upgrade** action when all required data maintenance is done.

The third option **Update**, which is considered outdated and is no longer recommended for use, does not remove anything from the target environment. This option just adds new components.

There are also different *API methods* that will be discussed in *Chapter 8, Microsoft Power Platform Extensibility*, to provide the possibility to automate the solution import procedures.

Other topics that have an impact on an ALM strategy in a Power Platform implementation are how Microsoft updates the platform, the types of updates, and the impact on the solution.

## Microsoft updates

To ensure the overall governance and consistent environment configuration, it is also necessary to consider the Microsoft-provided updating procedures of the Microsoft Power Platform and first-party applications (*Dynamics 365 applications*). Ideally, the whole environment's ecosystem for solution development should run on the same platform version. It is important to understand what updates are available and how to apply them. Basically, Microsoft provides *three* types of updates, as described in the following sections.

### Major Power Platform updates

The platform updates are provided bi-annually (*Wave 1 and Wave 2 releases*) and are applied automatically by Microsoft, accompanied by a lot of upfront, publicly available information and notifications. A customer cannot decide whether to accept or reject these updates. The only possibility to prepare for a major update is to prepare a dedicated testing environment, install the preview of the upcoming major update, and test the new capabilities upfront. The previews are usually available a few months before the general availability of the major platform update.

### First-party applications updates

First-party application updates are provided by Microsoft several times during a year but not on an exact schedule. In contrast to the major platform updates, these updates must be applied **manually** in the **Power Platform administration center**.



### Important note

A new option is provided in the admin center (as of writing this book, this option is in preview) to auto-update apps from selected publishers.

## Power Platform hotfixes

Hotfixes are unattended small updates provided by Microsoft very frequently and are installed automatically on every environment within a specified time frame. The hotfix announcements are made within the Microsoft 365 administration portal and mobile apps as knowledge base articles.

This concludes the detailed overview of how solution management works within Power Platform. We have discussed the overview of solution management, the types of solutions, dependencies, and segmentation, as well as the upgrade types Microsoft provides on the platform itself.

In the next section, we are going to deal with the Microsoft Azure DevOps service and its use for automating Power Platform deployment processes.

## Introducing Azure DevOps for the Power Platform

Azure DevOps is a Microsoft ALM solution for general-purpose software development, as presented in the previous chapter. With **Power Platform Build Tools**, Azure DevOps evolved into a mature tool for Power Platform solution development purposes as well.

Before Microsoft published Power Platform Build Tools, there were some community tools for Dynamics 365 available in the Visual Studio Marketplace that could also be considered since they have broad Dynamics 365-related capabilities.

In this section, we will present the capabilities of Power Platform Build Tools and the typical use cases in a Power Platform solution development.

## Overview of Power Platform Build Tools

Power Platform Build Tools is a collection of pipeline tasks. It is divided into four groups: **helper tasks**, **quality check tasks**, **solution tasks**, and **environment tasks**. We will discuss these groups in the subsequent sections.

## Helper and quality check tasks

The following helper tasks are part of the package:

- **Power Platform Tool Installer:** This needs to be the first task in every pipeline to ensure that the Power Platform Tools supporting artifacts, required by the DevOps agent, are installed properly.
- **Power Apps WhoAmI:** This task can be used to test the availability of a particular environment before a task with a transaction against that environment is triggered.

There is only a single quality check task, **Power Platform Checker**, which is used to trigger a quality check of a solution package using the *Power Apps Checker tool*.

## Solution tasks

**Solution tasks** can be used for various manipulations of the Power Platform solution files. The tasks currently require the *solution package file* to exist in the source environment. So, at the time of writing, there is no support for automatically creating solution packages. The following tasks are currently available:

- **Power Platform Export Solution:** Exports an existing solution from an environment as unmanaged or managed
- **Power Platform Import Solution:** Imports a solution package into an environment
- **Power Platform Apply Solution Upgrade:** Upgrades a solution that was imported as staging
- **Power Platform Delete Solution:** Deletes a solution from an environment
- **Power Platform Unpack Solution:** Unpacks a managed or unmanaged solution into the files contained in the package
- **Power Platform Pack Solution:** Packs solution files into an unmanaged or managed solution package

- **Power Platform Set Solution Version:** Sets a version number of an existing solution in an environment
- **Power Platform Deploy Package:** Deploys a package created with the *Package Deployer Tool* into an environment
- **Power Platform Publish Customizations:** Publishes all customizations in an environment

The solution tasks are key when creating deployment automation between Power Platform environments.

## Environment tasks

Environment tasks can be used for various manipulations of the Power Platform environments. The following tasks are currently available:

- **Power Platform Create Environment:** This can create a new environment. It is possible to select the region, target version, environment type, base language and currency, domain, and user-friendly name.
- **Power Platform Backup Environment:** This can create a backup of an existing environment.
- **Power Platform Copy Environment:** This can create a full or minimal copy of an existing source environment into a target environment. The target environment must exist; it cannot be created as part of this task.
- **Power Platform Delete Environment:** This can delete an environment.

The Power Platform Build Tools task types, together with the standard capabilities of Azure DevOps, can be used to implement several automations for a *Power Platform ALM approach*, as described in the next section.

## Azure DevOps with Power Platform Build Tools

There are several standard ALM scenarios in a Power Platform solution development that can be automated using the described tooling. In this section, we will discuss some of them.

## Committing a solution to the source control

One of the most common use cases in software development is **committing source code to a source control system**. While, in pure software development, this can be easily achieved with **Microsoft Visual Studio** integrated with *Azure DevOps* or any other source control system, in Power Platform with the solution management capability, this needs more effort. It is not recommended to commit unpacked solution packages into a repository directly, since a committed ZIP file cannot be used (for example) to compare versions. A typical solution commit procedure for an individual developer or for a consolidation environment could look like the following:

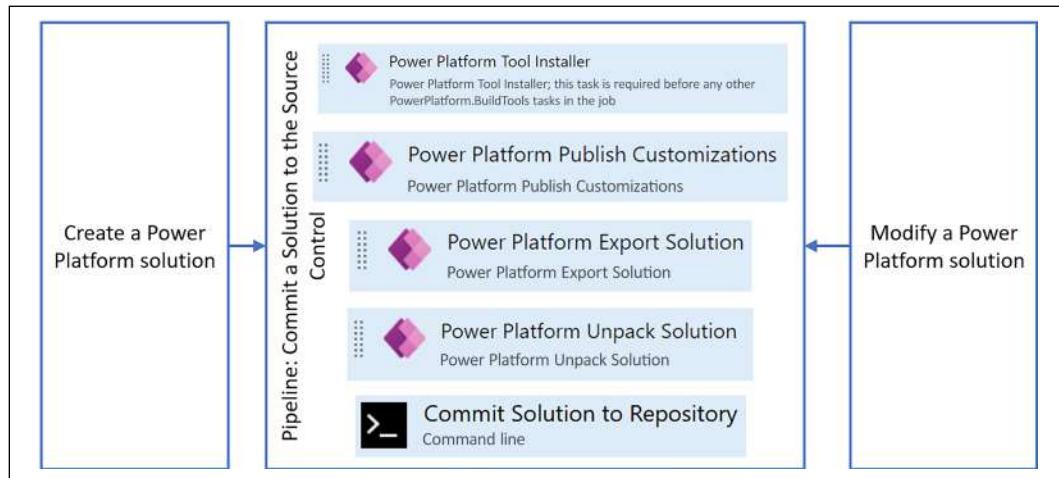


Figure 5.5: Committing a solution to the source control

For this scenario, first, a solution must be created in an environment manually, and then the **Commit a Solution to the Source Control** pipeline can be triggered after every modification of the solution content. In the pipeline, first, the customization should be published, then the export step should export the solution file as *unmanaged*. In the next step, the exported solution is unpacked. For the last step, committing the unpacked solution into the Azure DevOps repository, there is no Power Platform-specific task type, so the step must be implemented using scripting. For example, users can commit the artifacts into Git using *command-line procedures*.

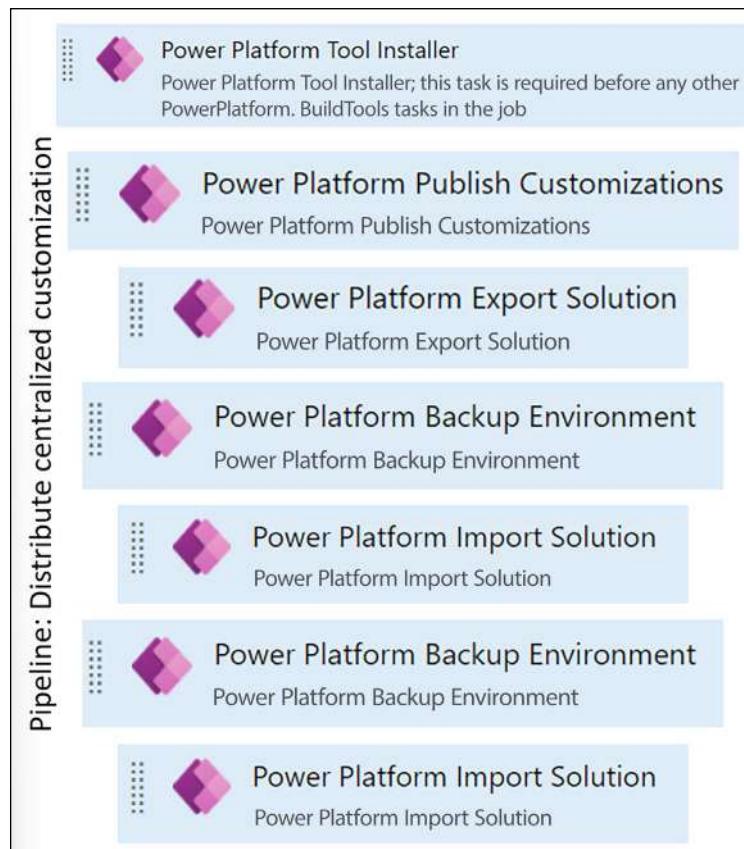
## Distributing solutions between development environments

This use case needs to be implemented for development teams where several team members are working on several development environments and there is a need for distributing the partial customizations between environments. There are two main scenarios for how this type of development can be organized. For both, the exported solution would always be unmanaged:

- **Centralized customization:** Centralized customization on a dedicated environment with distribution of the latest customizations to the development environments, where the developers are only performing custom development
- **Distributed customization:** Distributed customization and development on dedicated development environments, where the partial customizations and developments must be consolidated into a consolidation environment for further distribution

Both scenarios can be supported by solutions, as shown in *Figure 5.6*.

In the first scenario, there would be one pipeline distributing every change in the centralized customization to every development environment. The number of environment backup and solution import tasks would correspond to the number of development environments. The pipeline would be triggered after every change in coordination with the development team:



*Figure 5.6: Distribute centralized customization*

In the second scenario, there would be multiple consolidation pipelines or a single parameterized pipeline to transfer a partial customization from a particular development environment to the consolidation master environment. The pipelines would be triggered in a scheduled way or in coordination with the development team:

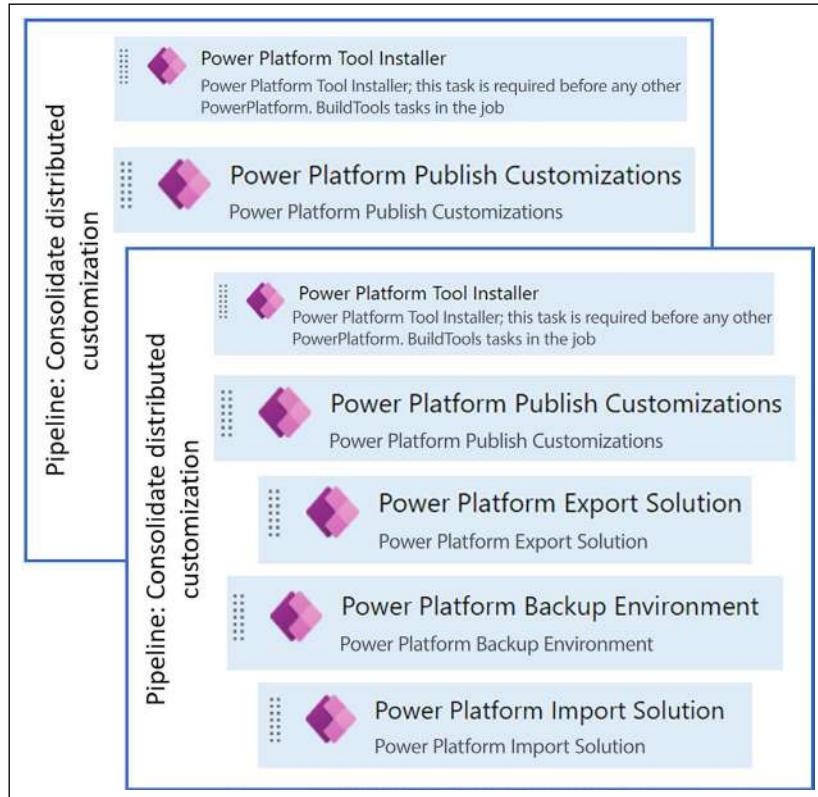


Figure 5.7: Consolidate distributed customization

In both scenarios, there is always an environment backup task preceding the import to ensure the target environment can be restored in case of any issue after the solution import.

## Distributing solutions out of development

This use case needs to be implemented when the consolidated solution should leave the development environments and be deployed into the downstream testing cascade as a managed solution:

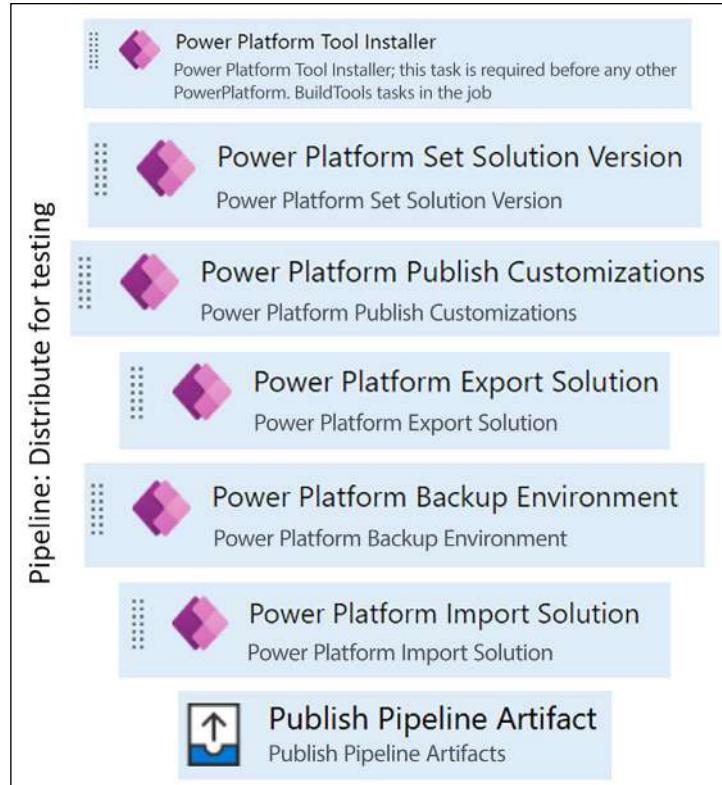


Figure 5.8: Distribute for testing

In this scenario, there are the following steps:

1. The consolidated solution usually gets a new solution version number.
2. The solution is exported as **managed** from the consolidation environment.
3. The solution is imported into the testing environment.
4. Optionally, there could be a task to publish the solution as a downloadable file using the Azure DevOps standard task, **Publish Pipeline Artifact**.

As always, there is an environment backup task recommended before the solution import step.

## Pipeline versus Release

Azure DevOps supports both the **Continuous Integration (CI)** and **Continuous Deployment (CD)** principles. The tool provides for this purpose the **Pipeline** component (representing CI) and the **Release** component (representing CD) for the automation processes. Based on the solution complexity, a decision needs to be made as to whether it is sufficient to use only the pipelines or to also implement the releases, which provide additional capabilities compared to pipelines.

Using Azure DevOps with the described Power Platform Build Tools is a great way to increase the reliability and predictability of the whole deployment process when building Power Platform solutions.

There is also another way to automate the deployment processes: using GitHub. We will describe this in the next section.

## Introducing GitHub for the Power Platform

GitHub supports similar ALM scenarios for the Power Platform as Azure DevOps does. In GitHub, there is a possibility to use a collection of Power Platform-related actions with similar functionality as the Power Platform Build Tools in Azure DevOps. In GitHub, the workflows configured using the workflow definition files (`.yml`) are used to implement various automations. By including the Power Platform GitHub actions in the workflow definitions, it is possible to automate the solution and environment-related processes in the same way as in Azure DevOps.

The Power Platform actions support the following capabilities:

- **Helper tasks:** There is just a single task, `whoAmI`, which is able to verify the service connection to a Power Platform environment.
- **Solution tasks:** In this group, there are tasks for exporting and importing, packing and unpacking, cloning, checking, upgrading, and publishing solutions.
- **Administration tasks:** In this group, we can find tasks for creating, copying, backing up, deleting, resetting, and restoring Power Platform environments.

Further, there are tasks handling packages and portal management.

### Important note



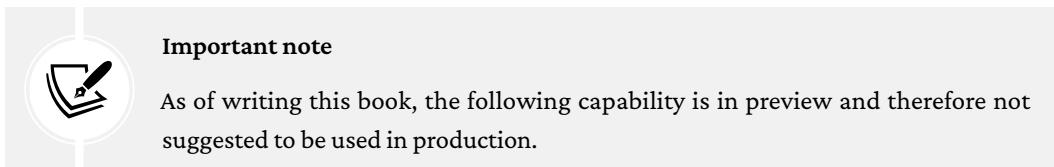
Please refer to the respective documentation to learn more about the Power Platform GitHub actions: <https://learn.microsoft.com/en-us/power-platform/alm/devops-github-actions>

Generally, with the GitHub instrumentation, we can build similar ALM solutions as with Azure DevOps.

In the following section, we are going to highlight the specifics of ALM for Power BI.

## Application lifecycle management for Power BI

As we have learned in the previous parts of this book, Power BI is technologically different from the rest of the Power Platform. Microsoft recently put a lot of effort into bringing the Power BI artifact closer to Power Platform, so that today, we can integrate the Power BI artifacts into Power Platform solutions to a certain extent.



### Important note

As of writing this book, the following capability is in preview and therefore not suggested to be used in production.

The following Power Platform components can be included in a Power Platform solution:

- Power BI report
- Power BI dataset

The Power BI components must be already imported into the Power BI service to be included in a Power Platform solution. The approach consists of the following steps:

1. Create a Power BI report in Power BI Desktop.
2. Publish the report into the Power BI service.
3. Install the Power BI Extension Package on a Power Platform environment with Dataverse.
4. Create a new solution or open an existing one and add the Power BI reports to the solution. The corresponding dataset will be added automatically.
5. You can equip the dataset with parameters to ensure, for example, that in a different Power Platform environment the dataset will be connected with the correct data in that environment.
6. As the report is included in the solution, the content (report and dataset) is exported from Power BI and imported into Dataverse.
7. For the imported Power BI components, a specific Power BI workspace is created, where these components are synchronized.
8. A set of security roles is created in Dataverse to support access to the Power BI components.

The above-described approach makes it much easier to integrate Power BI into the overall Power Platform ALM strategy.

In the following sections, we will discuss general ALM-related concepts for Power BI, not directly related to integrating/embedding Power BI content into the Power Platform solution management.

## Environments in Power BI

In Power BI technology, there is no such concept as Power Platform environments, so a different approach for ALM must be taken. It is recommended to use the concept of **Power BI workspaces** for this purpose, as illustrated in the following diagram:

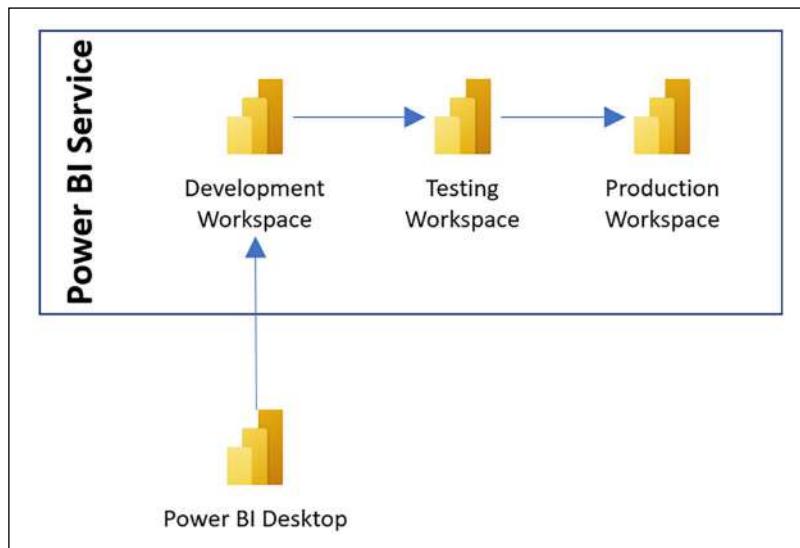


Figure 5.9: Power BI ALM workspaces

For every staging environment required for the ALM, a separate workspace can be created. The flow consists of the following steps:

1. A Power BI developer creates a report in the Power BI Desktop.
2. The report is published to the Power BI service, to a dedicated workspace for development.
3. Once the report is ready for testing, it is transferred to a different workspace, where the testing process can proceed.
4. The final Power BI report can be transferred to the production workspace to which the user group has access, to use the report.

Next, let us have a look at the Power BI components being handled in an ALM process.

## Power BI components

Despite the complexity of Power BI, there are just two component types that can be used in an ALM approach:

- **Power BI file (PBIX):** This component contains all the Power BI artifacts created with a Power BI Desktop, including the **datasets** with data.
- **Power BI template (PBIT):** This component contains all Power BI artifacts but **without** any data.

## Power BI ALM approach

It is necessary to also have an application lifecycle management option for Power BI, in other words, to be able to commit Power BI solution files to a source control system as well as automatically deploy the files between development, testing, and production. Therefore, we can use an approach such as the one illustrated in the following diagram:

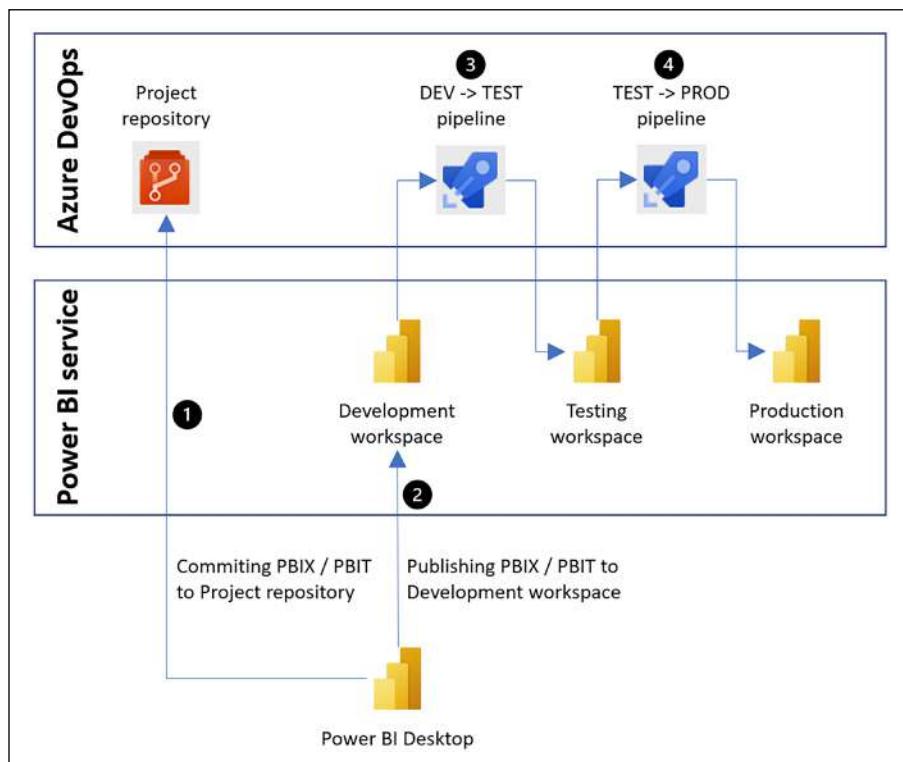


Figure 5.10: Power BI ALM approach

As we can see in the diagram, the respective components and processes can work in the following way:

1. The Power BI developers create Power BI solution components in Power BI Desktop and check them manually into an **Azure DevOps** repository as PBIX or PBIT files.
2. At the same time, the developers publish the solution components into the **Power BI Service** in the **Development Workspace** to test the functionality.
3. An **Azure DevOps** deployment pipeline exports the components from the **Development Workspace** and imports them into the **Testing Workspace**, where the testers can perform all necessary testing.
4. An **Azure DevOps** deployment pipeline exports the components from the **Testing Workspace** and imports them into the **Production Workspace** for production use.
5. The security settings in Power BI ensure proper access rights to all workspaces for the respective user groups.

The **Azure DevOps** pipelines can use PowerShell cmdlets or various community Power BI task packages, which can be found in the *Visual Studio Marketplace*.

The presented approach can completely cover the ALM requirements for Power BI and can automate most of the steps in a Power BI solution development process.

## Power BI deployment pipelines

For the Power BI Premium subscription, there is a new feature called **Power BI deployment pipelines**. This feature adds certain built-in ALM capabilities to Power BI, where it is possible to do the following:

- Create pipelines within the Power BI Premium environment.
- Assign workspaces to the pipelines.
- Upload Power BI content to the pipelines.
- Move content between pipelines (for example, from a Development pipeline to a Test pipeline, and finally to a Production pipeline).

It's up to the solution architect to decide which of the above ALM concepts to introduce in the overall Power Platform solution.

## Application lifecycle management for other solution components

Since a whole Power Platform solution can consist of many other solution components, the development of those components needs to be supported with ALM as well. More specifically, the following components need ALM support:

- Microsoft Azure solution components
- Custom development components for on-premises deployment

Since these components are developed with code, they can be easily managed by Azure DevOps. But they cannot be included in Power Platform solution packages, so the CI and CD must run separately from the Power Platform ALM automations.

## Application lifecycle management best practices

In the previous sections, we learned about the two key ALM pillars for Power Platform *solution development*: *solution management* and *ALM automation* using tools like *Azure DevOps*. In this section, you will learn several best practices for how to make the best of using these two components.

### Solution best practices

First, we will provide an overview of solution management-related best practices for how to properly use solution types, structure and segment solutions, and use publishers.

### General practices

It is the best practice to always create a specific solution package for a defined purpose and not use the **default solution** or just try to customize the environment outside of any dedicated solution package. While it is possible to include existing components in a solution package at any time, this should be only used to include standard components and never for custom components. Custom components should be always created in the context of a solution from the beginning. Strictly following this best practice will avoid the misconfiguration of components, for example, using the wrong prefix for custom artifacts or forgetting to include certain artifacts in a solution.

## Unmanaged versus managed

There is a clear purpose and very strong recommendations from Microsoft regarding the use of the proper solution type:

- **Unmanaged solutions:** These should be used **only** within the development environments, regardless of how many there are and the complexity of their structure.
- **Managed solutions:** These should always be used when the solution leaves the realm of the development environments and starts the journey to the downstream environments (various test environments, production, and so on).

The use of the proper solution type is one of the most controversial best practices, which is still not always respected, mainly for historical reasons, when the managed solutions approach was not mature enough and caused increased complexity and various issues.

## Structuring solutions

Deciding on the structure of solutions, especially for complex implementations, is always a complicated topic that does not have one single best answer.

Not using solutions at all is not an option unless we are preparing a quick demo or proof of concept. Depending on the complexity of the implementation, there are several ways to structure the solutions.

### Using a single solution

Using a **single solution** package for the whole Power Platform solution is a viable way for most of the situations where there are no dependencies on existing solutions or when the overall solution is not extremely large and the manipulation (export and import) with the solution package would not take an unreasonably long time.

### Using multiple solutions

Using **multiple solutions** always imposes a risk of creating collisions and dependencies and making the solution imports, exports, and deletions impossible. The only justification for using multiple solutions is when those solutions would be able to 100% avoid unwanted mutual dependencies.

Multiple solutions can be implemented in different typical situations, for example, when developing a complex Power Platform solution covering several workloads, and there is a need for specialist teams to develop the respective workloads independently from each other, as illustrated in the following diagram. But in such a case, it is also highly recommended to strictly define the customization boundaries for each specialist team to avoid conflicts:

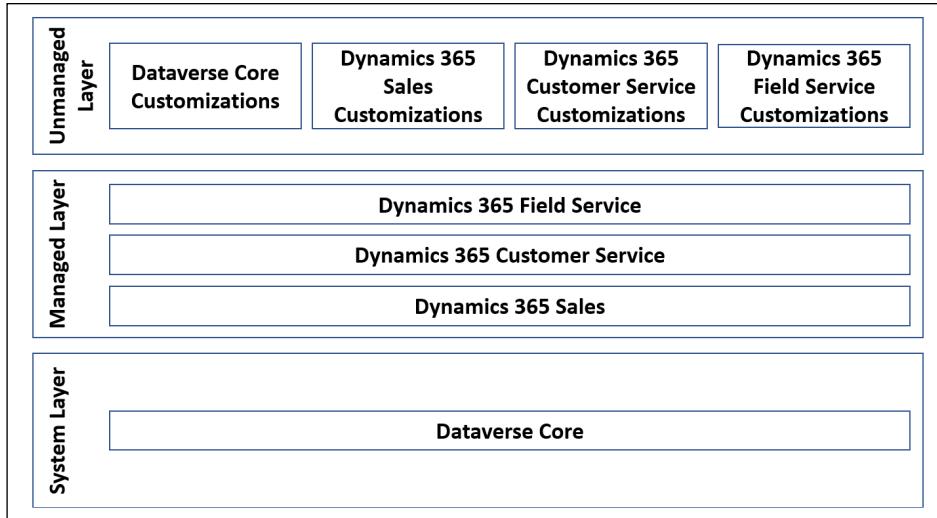


Figure 5.11: Workload-based multiple solutions approach

Another example of implementing multiple solutions could be for a **global solution** deployed into separate regional environments with certain per-region deviations in the user interface, business processes, and so on. This use case will intentionally create dependencies, but in this case, those dependencies are known and must be managed properly. An example of such an approach is illustrated in the following diagram:

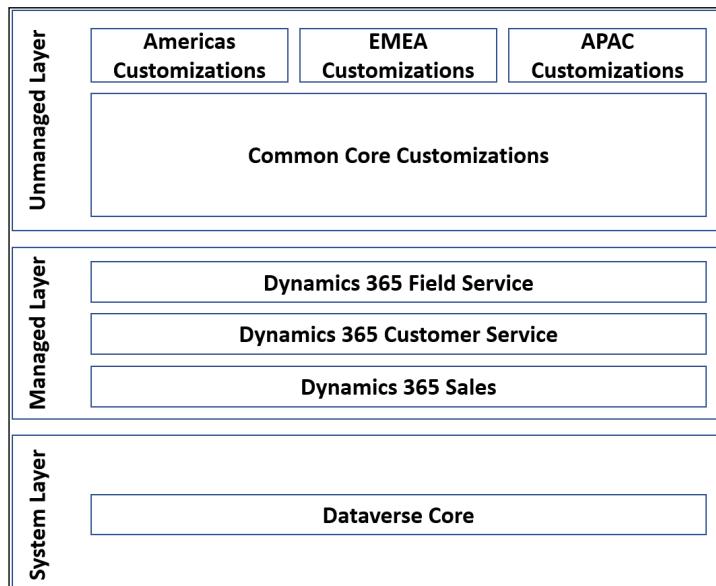


Figure 5.12: Regional-based multiple solutions approach

When using this approach, it is recommended to follow some best practices:

- Keep the custom functionality as much as possible in the **Common Core Customization**.
- Keep the **regional customizations** as small as possible.
- Do not implement any data model changes in the **regional customizations**, only additional UI elements and business processes.
- The regional customizations may have dependencies on the Common Core Customization but never in the other direction.
- Avoid any mutual dependencies between the regional customizations.
- Use solution segmentation to the highest possible extent to avoid collisions.

## Component sharing and component libraries

In the case that the implementation organization follows a strategy of creating libraries of **reusable components**, then it makes sense to place the reusable components into a separate solution package. These solution packages would usually have their own independent ALM. The whole management of these packages would follow product development best practices rather than a typical project implementation lifecycle. Such solution packages would need to be based on standard solution components only and be strictly independent of any custom artifacts in other solutions. To manage dependencies properly, such solution packages would need to be imported first to every environment where they would be used.

## Using segmentation

**Segmentation** can be used for Dataverse tables, as complex components containing subcomponents. When using segmentation, there are three options defining what can be included from the respective table in a solution, as illustrated in the following screenshot:



Figure 5.13: Segmentation options

It is important to understand what these three options do exactly and the best practices for using them:

- **Include all components:** This will add all components (metadata, columns, relationships, forms, view, charts, and others) of a table into the solution package. This option is used by default when creating new tables within a solution package. For existing tables, this option should only be used if that table is not yet present in the target environment.
- **Include entity metadata:** This will add only the table with its metadata, which is a collection of basic and behavioral settings for the table, but no table components. This option should only be selected when there is a need to change some of the table settings in the target environment.
- **Select components:** This is the true solution segmentation capability because clicking on this link opens a dialog where any table components can be individually selected and included in the solution. This option should be used for every situation when a modification of a particular component of a table should be deployed, for example, solution patching or solution updates.

## Using source control

The Power Platform solution package is a ZIP file, which is technically a binary file. Although it is certainly possible to commit this type of artifact into a repository, it would not be possible to use it within the repository for any further operations, such as content comparing. That's why it is highly recommended to always use the unpacking capability of the *Package Deployer tool* or the respective *Azure DevOps task* to first unpack the solution package into its components and commit the components into the repository. The solution package contains, for example, the whole customization configuration as a group of XML files and this can be very useful for analyzing customizations, comparing customization versions, and similar activities.

## Solution publishers' best practices

In the previous sections, we discussed several potential issues with collisions, dependencies, and so on, and presented various best practice approaches. The single most important best practice for the solution publishers to make all of the other best practices work is to always use **one single publisher for all solutions within a project**, regardless of how connected or isolated parts of the overall developed solution are.

## Power BI best practices

When developing Power BI components as part of an overall Power Platform solution, the following best practices can help to achieve more governance and consistency:

- Separation of datasets and visualizations, where the datasets can be created by data experts knowing the structure of the data model of the application, while the visualizations (reports and dashboards) can be created by more business-related experts
- Carefully structuring the required datasets to limit the amount of processed data and the size of the datasets
- Using pre-created templates for reports and dashboards (PBIT files) to ensure visual consistency of the created content
- Preferably using the PBIT files for ALM since they do not contain any data and are much smaller in size
- Using Power BI parameters in datasets to allow easy configuration of separate data sources for development, testing, and production

## Contoso Inc. ALM strategy

Contoso Inc. project and IT management analyzed the possibilities of ALM governance for their planned Power Platform solution implementation. They decided to implement strong governance from the very beginning and leverage all of the tools and possibilities available to help to organize a successful project delivery.

## Establishing Azure DevOps

Contoso Inc. confirmed the previous decision to establish Azure DevOps as the core ALM tool for all project activities and all project team members within the Power Platform solution development. The following was further decided:

- The *Power Platform Build Tools* will be installed on the Azure DevOps instance.
- An Azure DevOps project for the Power Platform implementation will be created.
- For all *Power Platform environments*, respective service connections will be configured within the Azure DevOps project.
- For all *ALM automations*, respective Azure DevOps pipelines will be implemented.
- All solution artifacts will be checked into an *Azure DevOps repository*.

## Using Power Platform solutions

Contoso Inc. decided to follow all recommended best practices for using Power Platform solutions:

- A common solution publisher **Contoso** with the prefix **contoso** will be used in all development environments.
- Unmanaged solutions will be used only within the boundaries of the development environments. For all other environments, strict use of managed solutions will be implemented.
- Solution versioning will be strictly followed.
- Solution segmentation will be used to the best possible extent to avoid collisions.
- Solution patches and updates will be used.
- Solution environment variables will be used for all environment-specific settings.
- A *multiple-solution approach* with separate solutions for the various workloads will be implemented. They have not yet decided about possible component sharing or using any common libraries or third-party components from the Microsoft AppSource.
- There will be no region-specific deviations from the central Power Platform solution.

The preliminary solution structure was specified according to the following diagram:

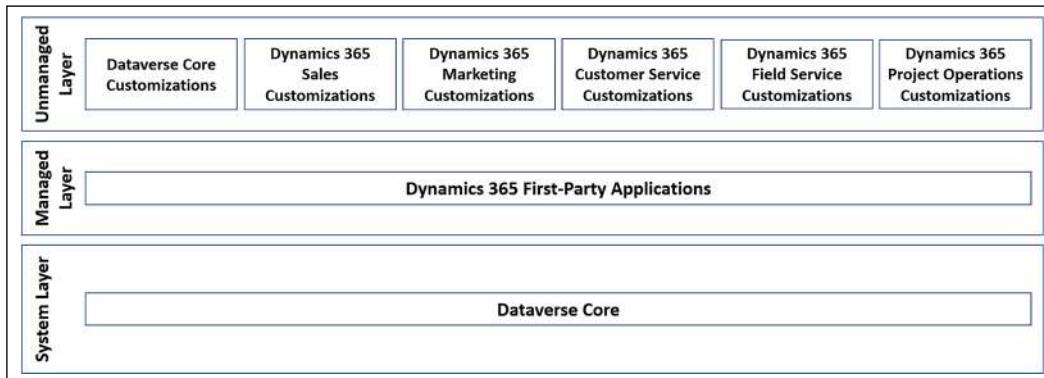


Figure 5.14: Contoso Inc. Power Platform solutions structure

Contoso Inc. decided that the project team needs to ensure that they completely avoid any overlaps of Power Platform components in the workload-related solution packages.

## Using Power BI ALM

Contoso Inc. already decided to purchase Power BI Premium to leverage the enhanced possibilities of its own dedicated Power BI capacity. They decided to use the recommended approach of separated Power BI workspaces and automated ALM procedures based on Azure DevOps.

## **Other ALM decisions**

All other components that will be part of the Power Platform solution will be developed using *Microsoft Visual Studio* or *Visual Studio Code* and integrated into the central Azure DevOps instance.

With these decisions, Contoso Inc. is now well prepared to start practical preparations for the Power Platform implementation project, as will be discussed in the next chapter.

## **Summary**

In this chapter, you have learned a lot about the tools, approaches, and best practices for establishing an *ALM strategy* for a Power Platform solution. This knowledge, together with that from the previous chapters, forms a foundation to start a real Power Platform implementation project.

In the next chapter, we will cover the practical aspects of an implementation project, with a focus on the implementation methodology and approach.



## **Share your thoughts**

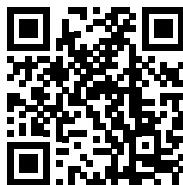
If this book is helping you improve your skills, we'd strongly suggest leaving a review on [Amazon.com](https://www.amazon.com). This helps us know if you like our work and if the chapter content has been valued, and also helps the buyers on Amazon know if the book is right for them.

So, everyone else benefits from your review - we wouldn't want you to miss out. You can now reach out to [review@packt.com](mailto:review@packt.com) with a screenshot of your review and the book URL, and we'll send you a \$5 voucher for your next Packt purchase. Thank you in advance for engaging with us, we are excited to see your review!

## **Learn more on Discord**

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/businesscenter>



---

# Section III

---

## The Implementation

After completing this part, you will have a full understanding of the implementation approach and methodologies, project setup and phases, project closure, and operations. You will also have an in-depth understanding of the security configuration and extensibility options of a Power Platform solution. Finally, you will have a full understanding of how a Power Platform solution integrates with a wide variety of Microsoft and third-party systems and solutions, and also be familiar with the challenges and best practices associated with data migration.

This section comprises the following chapters:

- *Chapter 6, Implementation Approach and Methodologies*
- *Chapter 7, Microsoft Power Platform Security*
- *Chapter 8, Microsoft Power Platform Extensibility*
- *Chapter 9, Microsoft Power Platform Integration*
- *Chapter 10, Microsoft Power Platform Data Migration*



# 6

## Implementation Approach and Methodologies

In this chapter, we will describe all aspects of software implementation projects with specific focus on the Power Platform. Implementing a Power Platform solution for a large international or global customer with an extensive number of detailed and complicated requirements can be a very complex and lengthy undertaking. It is very important to understand the recommended project *implementation methodologies* and typical *project setup* with roles and responsibilities as well as the whole *project life cycle*.

In this chapter, we will cover the following main topics:

- Overview of the project implementation methodologies and tools
- Project setup with the project structure, roles, and responsibilities
- Overview of the project phases, activities, and outputs
- Contoso Inc. project setup

### **Contoso Inc. preparing the implementation project**

Contoso Inc. has so far analyzed the Microsoft cloud offerings and decided to leverage the full potential of all of them to build a Power Platform business solution for the whole global organization. They have learned about the Power Platform architecture and the toolset necessary for building a solution, as well as the principles of application life cycle management in the Power Platform domain. As a next step, Contoso Inc. wants to learn about how to prepare and set up a Power Platform implementation project, what internal roles would need to be assigned, and what the expected project duration and phases are.

While they already have a lot of experience with running internal IT projects, with Power Platform, this is a new experience, and they would like to prepare everything for a successful implementation.

## Getting an overview of the implementation approach

There are many different IT implementation project types, from pure infrastructure projects to implementing robotics, artificial intelligence, or even a new CRM solution. While the value of a pure infrastructure project is mainly in areas such as infrastructure modernization, highly sophisticated IT projects must bring business value in the first place. Implementing a Power Platform solution is a typical project type focusing much less on infrastructure and much more on business value. The expected value of a Power Platform solution can be found in the following areas:

- **Modernization:** Replace the siloed, product-centric approach with a customer-focused approach (360° customer view).
- **Multichannel capability:** Extend the number of communication channels with the customer and provide an integrated omnichannel solution including social channels.
- **Automation and AI:** Extend the traditional data-centric manual processes with automations, artificial intelligence, mixed reality, and more.
- **Mobility:** Empower mobile workers with the same capabilities as traditional desktop workers.
- **Cost saving:** Change the traditional *CapEx* model to *OpEx*, saving significant upfront costs of a traditional IT solution, such as new hardware, software licenses, or even the extension of the own data center space.
- **Flexibility:** Provide a flexible and adaptable solution, being able to quickly respond to changing demand.
- **Agility:** Empower citizen developers to bring agility into the organization and reduce the dependency of IT by being able to participate in building IT solutions on their own.

All of these aspects and much more must be considered when planning a Power Platform implementation. The project is usually driven by business stakeholders, and they expect to get business value fast and do not think much about IT-related issues.

Here are some recommendations for a Power Platform implementation approach:

- Focus on business value, not on technology.
- Select a project implementation methodology that brings value quickly.
- Involve the customer in the solution development from the very beginning.

- Structure the project to provide some quick wins to motivate the customer.
- Help the customer with adoption and change management.

After this overview of the specifics of implementing a business solution, let's analyze what we can expect when we start talking to a company about implementing such a solution within their organization.

## Understanding customer enterprise architecture and environment

One of the key components to consider when planning a Power Platform implementation is the customer's own enterprise architecture. There is almost never the luxury to implement a *greenfield solution* for a new organization that has no existing IT ecosystem, no dependencies, no integration points, no legacy applications, and no established IT policies. A Power Platform solution will always need to integrate into the existing ecosystem and work well inside of it. That's why it's very important to understand the enterprise architecture and include it in the high-level architecting and designing from the very beginning. Here are some enterprise architecture considerations to evaluate:

- What is the organization's level of affinity to Microsoft technology in general?
- Is the organization prepared to accept a cloud solution?
- Does the organization require a cloud solution to reside in a certain geographical region or country?
- What kind of authentication ecosystem is established? Is it **Azure Active Directory** or something different?
- What are the IT security policies?
- Does the organization allow mobile workers?
- What are the data protection policies?
- Would the organization possibly require a data protection solution that cannot be implemented using a *public cloud approach*?
- Is there any approved and followed cloud integration strategy?
- What kind of legacy systems are operated in the organization? Do they offer any kind of API?
- Is the organization using a middleware for enterprise application integration? Is this solution compatible with Microsoft technology?
- How consistent are the business processes across the organization? Are there major deviations in different regions? Is the organization able to consolidate business processes?

- How many different languages should the solution support? Is the organization prepared to translate the Power Platform terminology and the terminology of the Dynamics 365 apps and their extensions into all required languages?
- What is the planned user distribution across the globe? Are there any business and non-business hours? What times might be suitable for running batch jobs?
- What is the level of maturity of the organization's adoption and change management? Are they able to ensure the smooth adoption of a major new IT solution?

All of these questions and certainly many more will influence how a Power Platform solution could be implemented, prompting us to consider what areas are causing increased efforts, what the major project risks are, and how to mitigate them.

Let's look at a few examples in the next sections.

## Data residency requirements

The organization requires the cloud solution to be located in **Sweden**. This is currently not possible since the Microsoft cloud region for **Sweden** is not yet Power Platform enabled.

The solution is to temporarily select some other Power Platform-enabled regions within the EU, for example, **Europe**, **Germany**, or **France**. As soon as **Sweden** is enabled for Power Platform, a move from the temporary region into the **Sweden** region can be requested from Microsoft customer support. This will impact the project schedule and impose certain additional effort since a move of environments to another region will also change the URLs of the environments. In addition, in a newly enabled Power Platform region, it is usual that not all Power Platform services are available from the beginning.

## Authentication providers

The organization does not rely on *Active Directory* as their main authentication provider but uses a *third-party solution*. This might make it impossible to use any kind of integration with *Azure Active Directory* and have an integrated single sign-on experience for the users of the Power Platform solution.

This situation could require additional efforts to implement advanced security within *Azure Active Directory* alone and to enable a cloud-only single sign-on capability.

## Internet restrictions

The organization does not allow mobile workers access to the public internet from their mobile devices, only to corporate network resources using a VPN.

To resolve this, additional effort would be necessary to ensure internet access for the Power Platform mobile applications routed through the mobile **VPN solution**.

## Data protection requirements

The organization does not trust the **Microsoft encryption technology** used in the Power Platform databases (Dataverse database) and requires using its **own encryption keys**.

This will require first becoming eligible for a **Bring-Your-Own-Key (BYOK)** solution and then planning to set up a solution using the organization's own encryption keys.

As you can see from the previously given enterprise architecture considerations and some practical examples, understanding the enterprise architecture of an organization and being able to offer suitable technological and commercial solution alternatives is one of the key success factors when planning a large Power Platform implementation project.

## Learning about project implementation methodologies and tools

In this section, you will learn about programs and projects, project methodologies, project effort estimation, tools used for Power Platform implementation projects, and the main project documentation types.

There is a multitude of different project management methodologies and many of them are not suitable for IT projects, so we will focus on the typical and most used Power Platform-related methodologies.

## Understanding programs and projects

It is important to understand the terms **program** and **project** and to be able to distinguish between them in the context of Power Platform implementation:

- A **project** is a temporary undertaking to create some product, service, or result.
- A **program** is a group of **related and mutually coordinated projects** to achieve a benefit that could not be achieved if they were not coordinated.

In the realm of Power Platform implementations, it is possible to consider a complex undertaking a large project but also a program if the parts of the project are very different, involving different parts of the organization and possibly implemented using different methodologies.

## Understanding project implementation methodologies

**Project management** is a specific and well-respected discipline, and using a proper and proven methodology is one of the key factors for successful project delivery. The IT industry is quite young but, in its short history, many project methodologies have been developed and adopted by the industry. The most well-known and IT-relevant methodologies and approaches are as follows:

- **Waterfall:** This is the traditional non-agile methodology. We will discuss the waterfall methodology later in this chapter.
- **Project Management Institute PMBOK:** This was adopted by Microsoft for their own *Sur-eStep 365* methodology dedicated to Dynamics 365 and Power Platform implementations.
- **Agile:** Agile, with modifications and extensions such as Scrum, Kanban, Scrumban, and Rapid Application Development, is used for small and fast-paced projects with very self-organized project teams.
- **Extreme Programming:** This is used for small and very fast and dynamic projects.
- **Adaptive Project Framework:** This is used for projects with a lot of unknowns to enable adjustments in scope.
- **PRINCE2:** This is popular for IT and other project types in the UK.
- **Information Technology Infrastructure Library (ITIL):** This is used for infrastructure projects.
- **Systems Development Life Cycle (SDLC):** This is a combined methodology that's popular for implementing both hardware and software solutions within a single project.
- **Iterative model:** This is a less dynamic methodology, where there is a mixture of waterfall and agile phases and where there is a typical overlap of the iterations.

Every software implementation project, regardless of the methodology or approach, has certain major phases, which might contain sub-phases. These are the typical phases of a software implementation project:

- **Requirements gathering:** The customer requirements are collected and documented
- **Analysis:** The collected requirements are analyzed and transformed into data models, business processes, and other artifacts
- **Design:** The results of the analysis phase are transformed into the technical architecture and design of the solution
- **Development:** The technical design is implemented into working solution components

- **Testing:** The solution components are tested and validated against the requirements
- **Deployment:** The finalized solution is transferred into production use

In the next part of this section, we will focus only on the four most widely used and adopted methodologies for implementing Power Platform solutions.

## The waterfall model

The **waterfall** approach is a traditional model where all the main project steps and activities happen sequentially, as illustrated in the following diagram:

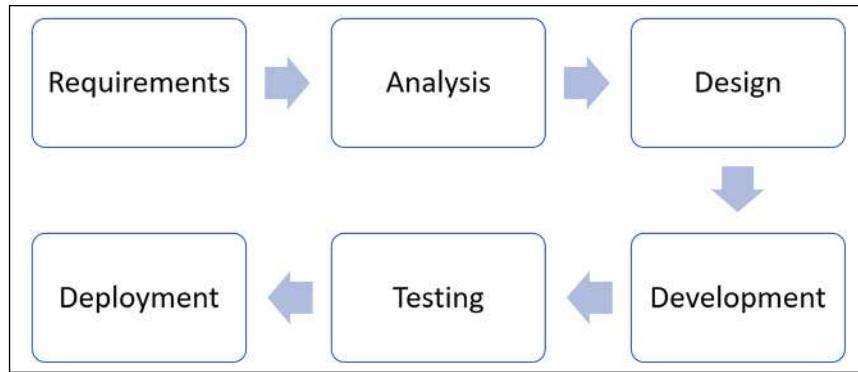


Figure 6.1: The waterfall project approach

The *waterfall model* has the following main features:

- The model is documentation heavy and very structured.
- The model is very role specific.
- Customer involvement during the project is limited.
- The model is not flexible in terms of responding quickly to changing requirements and situations.
- Testing happens usually after the full development is finished, which imposes the risk of **scope creep**.

### Important note



**Scope creep** is defined as uncontrolled change or increase of project scope. It can be caused by many reasons, one of which is lack of agility and customer involvement during the project. In a waterfall project the customer only sees the result after a long time of building the solution, which might lead to a large number of required changes.

For Power Platform, this project model is best suited for deeply technological and, at the same time, relatively independent parts of the program or project:

- **Infrastructure:** For on-premises deployments or for deployments with on-premises components
- **Integration:** For very complex integrations with various on-premises or third-party systems, with significant dependencies and implementation efforts on the legacy side
- **Data migration:** For very complex data migrations from various legacy systems with significant dependencies and implementation efforts on the legacy side

Generally, today, it is no longer recommended to use a waterfall model for implementing the business part of the solution since business requirements might change rapidly and the customer always expects to get business value quickly.

## The agile model

The agile model is a very dynamic one. It consists of a series of short cycles or sprints in which certain functionalities are analyzed, designed, developed, and tested, as illustrated in the following diagram:

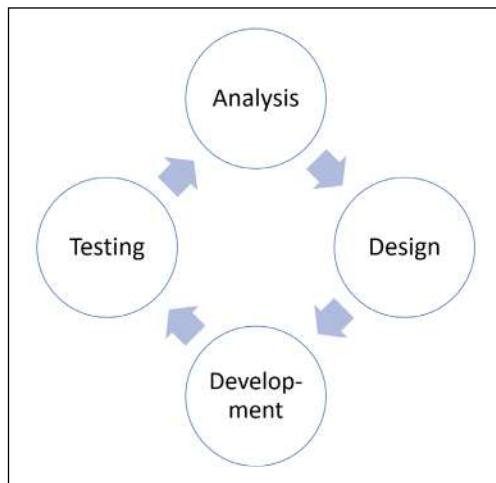


Figure 6.2: The agile project approach

The agile model has the following main features:

- The model uses light documentation.
- The sprints are short periods of time, usually two weeks.

- Agile teams are very self-organized. They decide the content of each sprint based on the overall situation and resource availability rather than by following a prescribed long-term plan.
- Team members can take any role; the model is not role specific.
- The customer is heavily involved in the project since they are part of the sprints.
- The model is very flexible and ongoing changes are easy to incorporate.
- Ongoing testing improves the overall quality.

For Power Platform, this project model is best suited to **small projects** with limited software development, integration, and data migration requirements and with few dependencies on legacy, on-premises, and infrastructure-heavy implementation requirements. At the same time, this model is suitable for dynamic projects with less detailed upfront scope definitions. It does not require heavy project documentation.

## The iterative model

The **iterative model** is less dynamic than the *agile model*. In this model, certain project phases are not part of the cycles, and the cycles (also called iterations) overlap. The model is illustrated in the following diagram:

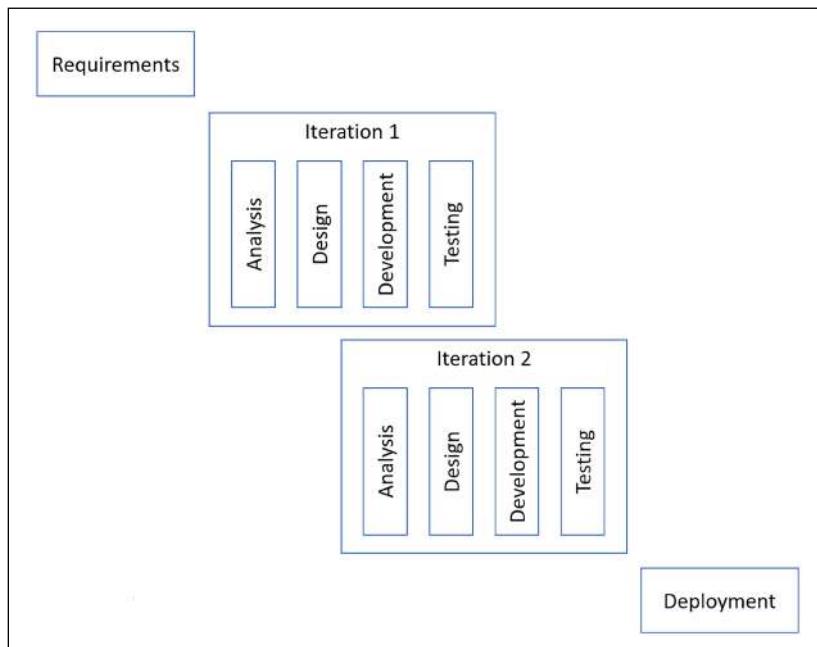


Figure 6.3: The iterative project approach

As you can see in the diagram, the iterative model has the following main features:

- The model usually has a non-iterative project starting phase dedicated to requirements gathering and a non-iterative final testing and deployment phase.
- The iterations are longer than those of the agile model, usually approximately four weeks.
- The iterations are overlapping so that the analysis and design of the subsequent iteration starts while the development and testing of the current iteration is still being executed.
- This model is more role-specific than the agile model.
- The customer is heavily involved in the project since they are part of the iterations.
- The model is flexible and ongoing changes can be incorporated.
- Ongoing testing improves the overall quality; however, there is usually an additional final non-iterative testing phase.

For Power Platform, this project model is best suited for **complex projects** with significant software development, integration, and data migration requirements and with dependencies on legacy, on-premise, and infrastructure-heavy implementation requirements. Furthermore, this model is best suited for customers requiring more project control and governance, more detailed and well-specified requirements catalogs, and more project documentation.

## The combined model

For large Power Platform projects or programs consisting of several projects, it is possible to leverage a combined model, where parts of a large project or individual projects within a program use different methodologies and models. Let's take an example of a Power Platform program consisting of the following three projects:

- A Power Platform **business solutions implementation** project following the **iterative** project model
- A Power Platform **integration** project to integrate with all existing legacy IT systems following the **waterfall** model
- A Power Platform **data migration** project to prepare and consolidate large datasets for initial data load following the **waterfall** model

Now that we have learned about the methodologies for a project, let's move on to another important task.

## Making a project effort estimation

**Project effort estimation** is a very important aspect of every software implementation, and it is important to explain the basics of estimation before we proceed to the next topics in this chapter.

At some point during the preparation of a software implementation project, there is a need to start working on an effort estimation. This estimation is very important for assessing the costs of the project, the project plan, and the schedule. There are two basic types of effort estimates:

- **Rough Order of Magnitude (ROM) or budgetary effort estimate:** This estimate can be created early in the preparation of a project. It is a rather high-level, interval-based estimate (for example, estimating the effort with an accuracy range of 75%–125% of the estimate) based on limited high-level requirements for the solution.
- **Final or firm effort estimate:** This estimate can be created when there is enough detailed requirement information from the customer to be more precise and improve the reliability of the estimate.

For a Power Platform solution, due to its nature, the effort estimate must be created in several different categories, covering the typical solution areas. We will discuss these categories in the following sections.

## Business requirements

The **business requirements** are usually estimated using a predefined business scenario or use case catalog or by specifying individually each business scenario, use case, or business requirements category. The efforts are estimated for the whole duration of the project or for the release in which they will be implemented using a complexity level and a *split formula* for the respective project phases, as in the following example:

Requirement	Complexity	Total hours	Analysis	Design	Development	Testing
			15%	20%	40%	25%
REQ-01	Simple	40	6	8	16	10
REQ-02	Complex	120	18	24	48	30

Figure 6.4: Business requirement example table

In the preceding example, a requirement of the complexity level **Simple** is estimated as **40 hours** total effort, while a requirement of the complexity level **Complex** is estimated as **120 hours** total effort. The split into four project phases is done automatically using the split percentages in the second line of the table.

## Custom development

When analyzing the customer requirements, an experienced architect should be able to identify gaps in those requirements and match them with the capabilities of the standard Power Platform and Dynamics 365 solution components. The gaps must be assessed and when there is a need for **custom development**, this effort needs to be estimated separately using a similar approach to that explained in the previous section. The only difference is, instead of business requirements, there would be **custom development requirements** along with the proposed technology and complexity level.

## Infrastructure requirements

If the Power Platform solution requires any **infrastructure** to be provisioned, these efforts must be estimated as well. The estimation for such a situation can be very individual, not following any specific methodology. Since some infrastructure might be necessary for the beginning of the project, some might need to be provisioned during the project execution. Infrastructure requirements usually impose important dependencies as the provisioning might not be completely under the control of the project. For example, some hardware, software, or cloud licenses might need to be purchased from hardware and software vendors. Any infrastructure requirements must be taken into account very early in the preparation of a project.

## Integration

Every complex Power Platform solution needs to be **integrated** into the customer IT environment. In fact, in many Power Platform solutions, the integration part is the most complex due to the nature of the integration, the technological complexity, and the dependencies on various third parties. The integration needs to be estimated very carefully to avoid budget overrun or other project schedule and cost issues during project execution. There are the following possible integration requirements:

- **Security integration**, such as Active Directory federation and integration with an identity and access management solution
- **Standard integrations**, such as integration with Microsoft Exchange and Microsoft SharePoint
- **Integration with other Microsoft cloud services**, such as with Microsoft Azure services
- **Integration with any ISV solutions** for Power Platform, acquired from Microsoft AppSource or third-party vendors
- **Integration with third-party public cloud services** such as Facebook and Twilio

- **Custom backend integrations**, such as integration with a legacy ERP system or other on-premises/cloud IT systems using direct integration or integration middleware
- **Custom frontend integration** with any possible existing IT application

There is no easy or unified way to estimate integration requirements. The best way is to perform an analysis of every required integration point and evaluate the technological complexity, dependencies, solution approach, risks, and other factors first. Then, assess the integration points with some complexity levels and get the effort estimate from these levels. The total effort needs to be distributed across the project phases in a similar way to business requirements. It is worth mentioning that a lack of an approved integration strategy for cloud solutions can increase the necessary effort since an alignment with many organizational units would need to happen before an integration can even be designed.

## Data migration

Every complex Power Platform solution contains a **data migration** or **initial data load** part. It is important to understand that data migration or initial data load is always a complex part of a project and should not be underestimated, as we observe quite frequently in Power Platform projects. It is equally important to realize that data migration cannot start at the end of the project but must be an integral part of the project from the very beginning. For a proper effort estimate of the data migration, the following information must be considered:

- Will the source system be integrated with the Power Platform solution or replaced with the solution?
- What kind of technology provides the source system?
- Is there easy access to the source systems?
- Is the structure of the source data known and documented?
- Are there subject matter experts available?
- What is the quality of the source data?
- Are there any dependencies between data from several sources?
- Into how many Power Platform data tables should the data be migrated?
- What is the overall volume of the data?

The best way to estimate the effort required for data migration is to consider the estimation at the individual table level. Consider the preceding list and assess every table with a complexity level and get the effort estimate from the complexity level. The total effort needs to be distributed across the project phases in a similar way to the business requirements.

## **Other efforts**

There are always some additional efforts that need to be estimated individually and included in the total estimate in the correct project phase. Here are some examples of those efforts:

- **Initial project phases** such as project preparation, initiation, and initial analysis
- **Final testing** such as **User Acceptance Testing (UAT)**
- The **solution deployment phase** covering the final steps of the project, such as the deployment of the accepted solution packages to production environments and the final data migration
- **Localization**, which needs to be part of the solution development for multilingual solutions
- **Training**, which is usually planned for the last project phases
- **Go-live support**, which is usually contractually covered for a certain time after the solution is finally deployed

All of these efforts are usually estimated individually based on solution complexity, customer requirements, collective experience, and other factors.

## **Project management tools**

There are some important project management tools that can be used to manage Power Platform projects. In this section, we will briefly describe the main capabilities of Microsoft Project and Azure DevOps and focus on the topic of effort estimation.

### **Microsoft Project**

Microsoft Project is a group of product management tools. We will explore these tools and their main features in the following sections.

#### **Microsoft Project desktop client**

The **Microsoft Project desktop client** is a member of the Microsoft Office family of products and is primarily used by project managers to create and configure new projects, create work breakdown structures, and other routine project management tasks. The client can be connected to Microsoft Project Online to enable project team member access.

#### **Microsoft Project Online**

**Microsoft Project Online** is a SharePoint-based project management backend that can be used specifically to track the work of project team members for projects created and configured by project managers.

## Microsoft Project for the Web

**Microsoft Project for the Web** is a new Power Platform-based project management solution. The product can be used for similar purposes to Project Online; however, there is no native integration with the Project desktop client. Since this product is based on Microsoft Dataverse, it can easily be customized and tailored to the needs of the customer.

The Microsoft Project family of products can be used to manage waterfall as well as agile and iterative Power Platform projects. The benefit of using Microsoft Project is in the initial phases of a project for creating high-level project plans and resource allocations as well as for managing waterfall projects that are not directly supported by Azure DevOps.

## Azure DevOps

Azure DevOps has very broad capabilities. Besides the capabilities for application life cycle management discussed in the previous chapter, the product also offers capabilities for project management of agile projects. It is possible to leverage the following capabilities:

- **Perform a general project configuration action** such as selecting the project process type (Basic, Agile, Scrum, or CMMI), configuring project teams and permissions, configuring sprints and assign them to teams, and configuring boards.
- Create, assign, and track various **work items** (epic, feature, user story, task, issue, bug, and others) depending on the process type.
- Track the work items on the interactive **backlogs** and **boards** across various workflow states (to-do, doing, done, new, active, resolved, closed, and removed) depending on the process type.
- **Collaborate** on work items using the discussion capability.
- **Plan sprints** and work in sprints.

While Microsoft Project and Project for the Web are better suited for strategic program and project management planning, Azure DevOps is better suited for the **daily** operational project management of agile and iterative projects, since the tool is directly connected to the development process. Azure DevOps, however, does not provide any specific support for waterfall projects.

## Effort estimators

Compared to the previously mentioned tools, for the very important task of project effort estimation, there is no tool that can be explicitly recommended. Software consulting companies mainly use internal single-purpose estimators, in most cases based on Excel templates containing estimating rules based on collective experience and intelligence, as explained in the previous section.

## **Creating project documentation**

Depending on the project methodology or approach, part of the project delivery will be creating certain project documentation. In this section, we will describe the most typical project-related documentation used in Power Platform implementation projects.

### **The project plan**

A project plan is the main project management document and needs to be created in some form for every project. A project plan contains the following main information:

- Project start and end dates
- Project phases and duration
- Project activities within the phase
- Project milestones
- Project resource allocations

You can see this main information illustrated in the following example created with Microsoft Project:

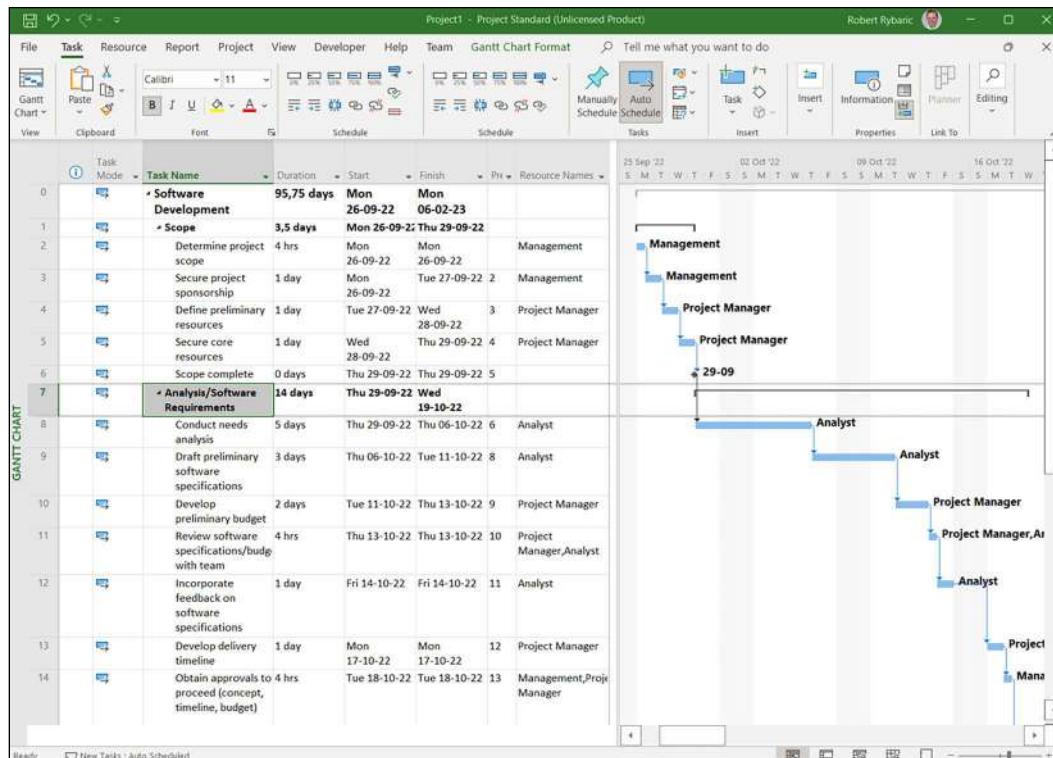


Figure 6.5: A project plan example

In the preceding example, there are certain project phases with sub-phases and project activities. For every project activity, there is a start date, an end date, the duration, and resource assignment. On the right side, there is a *Gantt chart* graphically representing the project flow.

## **The requirements document**

The requirements document usually has its source in the customer requirements catalog, which is used in the bidding and selection process. The final requirements document, however, is based on the project analysis phase, where all requirements are discussed with the customer, analyzed, consolidated, and documented. The requirements document is usually in a table form reflecting the following information:

- Requirement ID and name
- Requirement type (functional or non-functional)
- Further requirement classification
- Complexity
- Priority
- Fit-Gap assessment

The requirement document in its initial state is often used as a source for the project effort estimation. When finalized, the document is used as a foundation for solution architecture and design.

## **The solution architecture document**

The solution architecture document is a high-level document specifying the following areas:

- Solution architecture (overall solution composition)
- Infrastructure architecture (cloud and on-premises infrastructure components composition)
- Data architecture (high-level data model)
- Security architecture (authentication, authorization, and data protection)
- Integration architecture (overview of all integrated IT systems and high-level integration specification)
- Data migration architecture (consolidated view of all required data migrations)
- Reporting and analytics architecture (composition of the reporting components)
- Any other required architecture domain documentation

A more detailed technical document is created later during the project execution, known as the solution or technical design document, which is described in the next section.

## The solution/technical design document

The **solution or technical design document** goes into more detail than the solution architecture document by specifying the solution in the following areas:

- Data model design (a full data model of the Dataverse database covering columns, relationships, and any other database components used in the solution)
- User interface design (a design of model-driven apps, canvas apps, Power Pages portals, and reports)
- Security design (a detailed specification of all relevant security settings)
- Custom development design (a design of all custom-developed artifacts for the Power Platform components, integration, and data migration)
- Any other required design specifications

There are several other typical project documents, as described in the following section.

## Other documents

Depending on the complexity of the project and customer requirements, additional separate project documentation may be required. In the following sections, we will have a look at some of these documents.

## Testing documentation

Testing is a very important part of every software implementation project and requires specific documentation, mainly the following:

- **Test strategy:** This describes all components of testing within the project, types of required tests, responsibilities, and testing tools.
- **Test cases:** These describe every required test case in the necessary detail, specifically the inputs, test conditions, testing procedure, and expected results.

Another usual documentation type is documentation for training the users and administrators for the solution, as described in the following sections.

## **Training documentation**

For the purpose of formalized end user or key-user training, training documentation might be required containing the training approach and end user training material. The training material can be of different types:

- Training manuals
- Online web-based training material
- Training videos

The next important area you will learn about is how to set up a Power Platform project in terms of project roles and responsibilities.

## **Learning about project setup**

In this section, you will learn about the difference between internal and external project types and about project roles and responsibilities in a large external Power Platform implementation project.

## **Project types**

Every company has the freedom to decide whether they want to implement a new software solution using internal resources and capabilities only or to select and hire a professional services consulting company to perform the implementation.

### **Internal project**

The ability to run a software implementation project such as Power Platform internally is influenced by a number of factors:

- Internal experience of running software implementation projects

- Availability of internal resources that can be assigned full time to a project for several months or even years
- Existing expertise in the domain of the implemented software
- Costing and scheduling considerations
- Strategic interest to build up skills in the domain, which could be used later to deliver the services externally

## **External project**

If a company decides not to run the Power Platform implementation internally but rather appoint an external consultancy, the main consequences are as follows:

- There's no need to build up skills for an internal team up front.
- Less internal resources need to be allocated to the project.
- The project can be planned for an earlier start.
- The company will need to run a standard selection, bidding, and procurement process.

In the rest of this chapter, we will focus only on the typical case, where a Power Platform implementation project is delivered externally by a consulting company. Internal projects are rare since a lot of deep expertise and experience is needed to successfully implement such a solution.

## Project roles and responsibilities

The organization structure for Power Platform implementation projects can be anything from very simple to very complex. Simple projects can operate with few resources in a *flat hierarchy*, while a very complex project might need a *multi-step hierarchy* with a large number of resources from both the customer and the implementation partner. An example of a larger project setup is illustrated in the following project organization structure diagram:

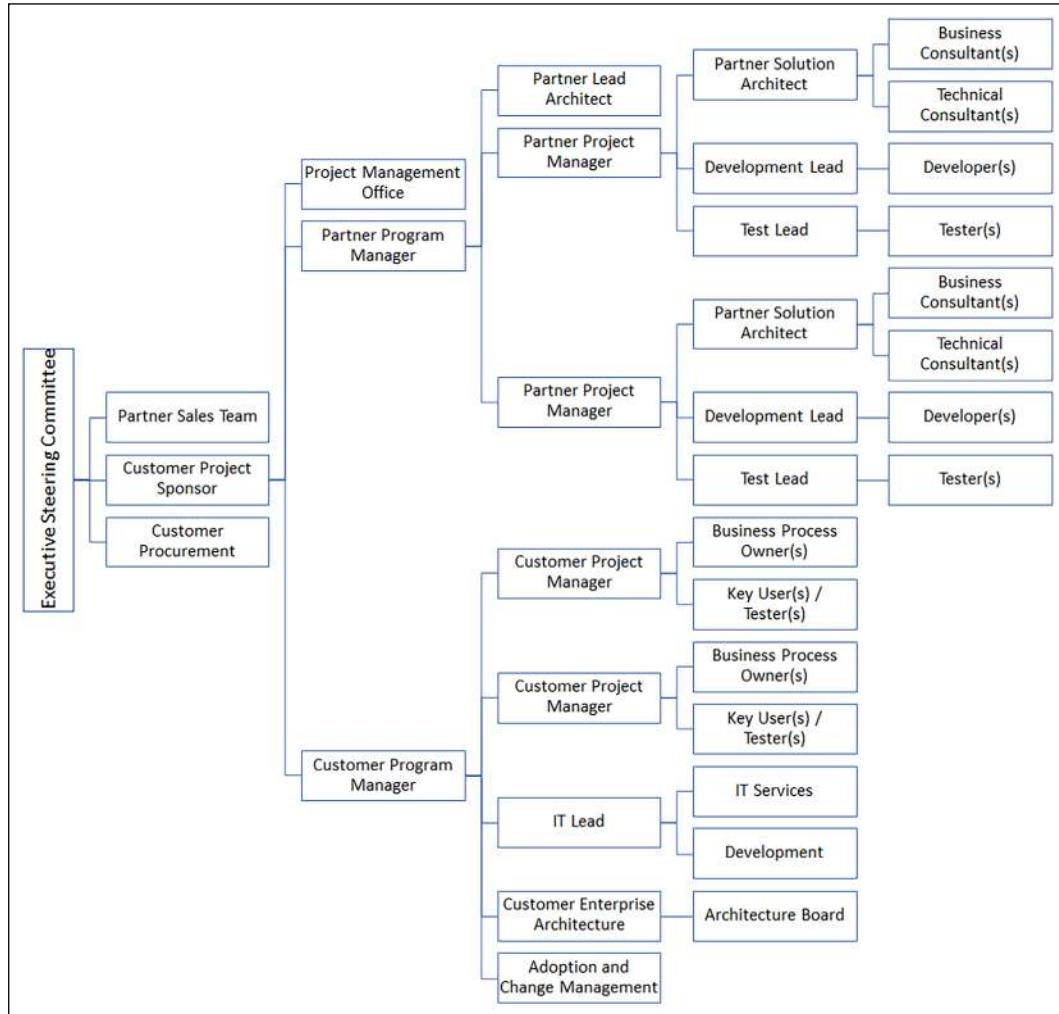


Figure 6.6: Complex Power Platform project organization structure

Figure 6.6 illustrates a project organization structure with six levels of hierarchy. This complexity is not unusual for large Power Platform implementations.

## Central roles and responsibilities

There are certain central roles in each project that have some key responsibilities but are not part of the project team, and their commitment to the project takes up just a small percentage of their time.

### Executive steering committee

The **executive steering committee** is a group of individuals representing higher management positions from all major organization units, senior professionals, and experts. The members should be able to steer all projects running in the organization to ensure that company policies and objectives are followed, resources are allocated, budgetary constraints are under control, and projects are in line with the overall organization strategy. The projects usually require the steering committee to make key decisions as they influence the whole organization.

### Customer project sponsor

The **customer project sponsor** is usually a representative from higher management who serves as an escalation level, makes key decisions, and ensures that major project issues and roadblocks are cleared.

### Partner sales team

The **partner sales team** is mainly involved during the preparation phase of the project, but they are also involved in all commercial topics during the whole project execution.

### Customer procurement

The **customer procurement team** is mainly involved during the preparation phase of the project but might also be involved to a certain degree during the execution to ensure the procurement policies are followed.

### Project Management Office (PMO)

The **PMO** is either a permanent department within an organization or a temporary team of individuals established solely for the purpose of supporting an ongoing program or project.

Permanent PMOs usually have competencies such as project management methodology standardization, project compliance control, and project documentation and guidance.

For a Power Platform implementation, it is rather a temporary program or project-related PMO that fulfills the supporting role for the project management (program manager, project managers). The PMO can serve the project in the following areas:

- Administrative support
- Support in project reporting and monitoring
- Support in project control

A project-related PMO can be established either on the customer side, the partner side, or as a mixed team with members from both organizations.

## **Partner roles and responsibilities**

The consulting company or implementation partner must provide a project team of experts to be able to deliver the required solution. Based on customer requirements and partner capabilities, part of the project team might work onsite at the customer premises, but the work of certain roles can also be performed remotely from partner offices, home offices, or offshore locations. The use of offshore project resources is very popular, specifically to achieve a competitive blended hourly or daily rate.

In this section, you will learn about the typical partner project roles and their responsibilities.

### **Program manager**

**Program manager** is an optional role only for situations where the solution implementation is subdivided into multiple projects. The program manager then manages all of the underlying projects from a high-level point of view; so, rather than managing daily activities, they take care of the overall coordination between the projects and the strategic direction and the overall value all of the projects bring to the customer. They usually coordinate between the project managers and activities within the PMO, if established. The typical activities of a program manager include the following:

- Creating a program management plan
- Specifying the program scope, schedule, and budget
- Managing the milestones of a program
- Managing high-level program risks, communication, scope, and quality
- Managing the program escalation process and the stakeholders
- Performing program closure

## Project manager

**Project manager** is a key role in every project, managing all aspects of the assigned project (or projects) on a daily basis. The typical activities of a program manager include the following:

- Creating and managing the project plan, schedule, and budget
- Managing project resources and organization
- Managing project communication, quality risks, issues, priorities, and changes
- Managing the project escalation process and stakeholders
- Managing project administration and reporting (with support from the PMO, if established)
- Performing project closure

## Lead architect

**Lead architect** is an optional role only for situations where the solution implementation is divided into multiple projects. In this case, the lead architect would take the following responsibilities:

- Ensuring consistency of the overall solution across projects
- Responsibility for the overall solution design across projects
- Responsibility for following best practices across projects

The responsibilities of a lead architect within a Power Platform implementation program can be the following, for example:

- Ensuring a unified Azure DevOps environment is established for all development teams in all Power Platform projects
- Ensuring a unified data model is developed across developed Dynamics 365 solutions and Power Platform integration with customer IT systems and a developed data migration solution
- Deciding whether the developed Power Platform integration with customer IT systems can also be reused for data migration
- Consolidating technical access to customer IT systems for all Power Platform projects

The lead architect is usually invited to be a temporary member of the customer's architecture board, if one has been established.

## **Solution architect**

**Solution architect** is a key role in every project, in terms of coordinating the solution development from the customer requirements and technology point of view, with the following responsibilities:

- Coordinating the translation of the business requirements into a solution vision
- Coordinating the creation of the overall solution architecture and design
- Selecting the best technological implementation approach
- Guiding the implementation team and ensuring best practices are followed

The responsibilities of a solution architect within a Power Platform implementation project can be, for example, as follows:

- Elaborating the solution architecture by deciding which customer requirements will be implemented using Dynamics 365, custom model-driven apps, canvas apps, and Power Apps portals
- Elaborating the solution design by deciding which automations will be implemented using Power Automate, Dataverse workflows, business rules, PlugIns, or Azure Functions, just to name some possible approaches
- Designing the integration architecture by analyzing each integration requirement and deciding whether the integration can be implemented using Power Automate, Azure Logic Apps, Azure API Management, Azure Functions, Azure App Service, Azure Service Bus, existing customer middleware, or custom development
- Designing the reporting architecture by deciding whether Power BI can be used for all reporting requirements if the Azure SQL database needs to be established to consolidate various data sources for reporting

A solution architect can lead the business and technical consultants. In smaller projects without assigned consultants, however, the solution architect can play the role of the consultants as well.

## **Business consultant**

A **business consultant** is responsible for analyzing and documenting business requirements and preparing them for technical design, participating in the creation of end user documentation, testing documentation, performing end user training, and other such activities. The responsibilities of a business consultant within a Power Platform implementation project can be, for example, as follows:

- Running customer requirements workshops

- Presenting out-of-the-box capabilities of the Power Platform components and document gaps
- Assisting in the creation of technical design documents
- Assisting in the customization of Dataverse, model-driven apps, canvas apps, Power Automate flows, and Power BI
- Preparing Power Platform test cases
- Preparing Power Platform training material
- Running train-the-trainer customer training sessions

## **Technical consultant**

**Technical consultant** is an optional role for situations where the consulting responsibilities should be split between business and technical consultants. This can be necessary, for example, due to the high complexity of the project or extensive specialization of the resources. The responsibilities of a technical consultant within a Power Platform implementation project can be as follows, for example:

- Creating technical design documents
- Customizing Dataverse, model-driven apps, canvas apps, Power Automate flows, and Power BI

## **Development lead**

Every complex Power Platform implementation project contains certain custom development efforts. To cover these efforts, it is necessary to include developers in the team. A development team in a project usually consists of a **development lead** and **developers**. The development lead has the following typical responsibilities:

- Coordinating custom development planning with the project manager and solution architect
- Coordinating all custom development activities in the project
- Ensuring overall development quality

## **Developers**

The **developers** in a Power Platform project implementation are responsible for writing code and performing all necessary testing of their own code (**unit testing**). If there is a large development team, the individual developers might work on individual tasks or even on large custom development artifacts as a group. In this case, the development lead is responsible for ensuring proper coordination and managing mutual dependencies between the individual developer's tasks.

## Test team

There is no quality assurance in a Power Platform project without testing. The testing is performed by a dedicated test team consisting of a **test lead** and **testers**. The test lead has the following typical responsibilities:

- Defining the project test strategy together with the solution architect and the customer
- Consolidating test scenarios created by business consultants or customer
- Coordinating all testing activities in the project

The **testers** are responsible for creating test scripts and performing all necessary types of tests.

## Other roles

In a complex Power Platform implementation project, any number of other specialized roles may be required that are not included in the project organization structure example illustrated in *Figure 6.6*, such as the following:

- **Infrastructure consultant:** This role is responsible for performing any infrastructure-related activities when they are part of the project (configuring hardware, installing and configuring software, and providing support for the customer's IT staff).
- **Release manager:** This role is responsible for application life cycle management activities (setting up Azure DevOps components such as the repository, pipelines, performing solution distribution, and other similar activities). This role can be also assigned to the development lead or test lead.
- **Integration lead:** This role is responsible for the development of the integration solution.
- **The data migration lead** is responsible for the development of the data migration solution.

## Customer roles and responsibilities

One of the key reasons why software implementation projects fail is a lack of customer commitment. Large companies, having experiences with IT projects, understand the need to provide enough resources and overall commitment for such projects. It is important to convince every customer that commitment is key; all necessary roles must be staffed, and all individuals assigned to the project roles need to be partially released from their daily responsibilities.

In this section, you will learn about the typical customer project roles and their responsibilities.

## **Customer program manager**

The responsibilities of the **customer program manager** are similar to that of the partner program manager, who is their close cooperation partner during the program. The program manager could be a temporary assignment, or they could be in a permanent role, for example, as part of the customer's PMO. A permanent program manager manages the changing portfolio or currently running projects in the organization.

## **Customer project manager**

The responsibilities of the **customer project manager** are similar to that of the partner project manager. It is recommended to mirror the program structure on both sides so that there is a separate customer project manager for every project within a Power Platform implementation program. Similarly to the customer program manager, the project manager can be a temporary or permanent role, depending on the organization's structure.

## **Business process owner**

**Business process owners** or business analysts are key project team members on the customer side since they are the subject matter experts for the business processes being implemented within a Power Platform solution. They have the following typical responsibilities:

- Providing expertise and advice for business processes in their functional domain
- Deciding on business process structures, functional details, data, user interface, workflows, and other elements of the implemented business processes
- Support in streamlining and standardizing business processes
- Support in developing test scripts

## **Key user/tester**

**Key users** or customer testers contribute to the overall quality of the developed Power Platform solution by being involved in the project from the beginning and providing constant feedback. They have the following typical responsibilities:

- Support in developing test scripts
- Testing every new feature as provided by the project in the sprints or iterations and providing feedback to the project

It is not uncommon for large organizations to have dedicated test teams with a test lead, which take some responsibility for testing the solution.

## **IT lead**

Every complex Power Platform solution needs to be heavily integrated into the customer's IT ecosystem. The **IT lead** of the organization, which can be any role from CIO to the data center manager, must be included in the project to ensure full support of the customer's IT for successful project implementation. The IT lead has the following typical responsibilities:

- The primary technical point of contact for the project
- Managing and providing IT resources for the project
- Providing support in developing the solution's technical architecture

## **IT services**

During a Power Platform implementation, various support services for the customer's IT organization are required. The following roles are usually involved in the project:

- **System administrators** are responsible for provisioning any necessary hardware or software components within the customer's data center.
- **Application administrators** are responsible for providing access to the existing IT applications involved in the Power Platform solution (integration and data migration). They provide support and subject matter expertise for the project team.
- **Database administrators** are responsible for providing access and supporting the project with existing database solutions involved in the Power Platform solution (integration and data migration).
- **Security administrators** are responsible for providing secure access to the customer's IT infrastructure.

## **Development**

As mentioned earlier, every large and complex Power Platform solution implementation is heavily integrated into the customer's IT ecosystem and includes large data migration efforts. The consulting partner must be able to cover all implementation efforts on the Power Platform side, but usually not on the side of the various customer's legacy systems. That's why certain custom development efforts must be provided by the **customer's own development resources** or including vendors with expertise in those areas. The customer's developers have the following typical responsibilities:

- Providing guidance to the partner's development resources regarding custom development against the own IT systems

- Developing or modifying interfaces for Power Platform integration and data migration
- Developing new integration scenarios on customer's integration middleware, if used
- Developing data migration scripts to export data from customer's IT systems
- Developing security solutions to integrate Power Platform security into customer's IT corporate security

## Enterprise architecture

Many large organizations have an **Enterprise Architecture (EA)** department or team that's responsible for the standardization and classification of all business processes, application processes, data structures, and information systems in the organization. The EA team is also always involved in all IT projects since the implemented solution must be compliant with the *EA standards* of the organization. The EA team is usually involved already in the project preparation by giving guidance on the best solution approaches from an EA point of view. During project delivery, the EA team ensures the project follows EA standards and guidelines.

## Architecture board

Part of the enterprise architecture unit in the organization is often an **architecture board**, which is a committee of IT managers, architects, and other experts providing oversight over all running IT projects in the organization. During a Power Platform implementation project, some key project team members, such as the lead architect, solution architects, and others from the implementation partner, are invited to be temporary members of the architecture board. The architecture board discusses and analyzes architectural concepts, changes, risks, and issues and makes key architectural decisions.

## Adoption and change management

Even though **Adoption and Change Management (ACM)** is a joint responsibility of both the customer and the implementation partner, there is often a formalized organizational unit within organizations to manage adoption and change activities related to all changes in the organization. It is important to understand that ACM needs to be incorporated into the implementation project from the beginning as one of the key reasons for the successful adoption of the new solution. Typical responsibilities of the ACM involved in a Power Platform implementation are as follows:

- Defining the change and success criteria
- Driving the change across the whole organization's hierarchy
- Properly communicating the purpose and benefits of the change
- Providing training opportunities for all key user roles of the new solution

- Collecting user feedback
- Measuring adoption success

After learning the details about the project setup for large Power Platform implementation projects, we will now analyze and discuss the typical Power Platform project life cycle with all possible phases from the initial concept until a solution is brought into daily operation.

## **Understanding project phases**

At a high level, a project consists of three main phases, and each phase can consist of many sub-phases. Not all of these sub-phases are always present or in the same order as described in the following sections, but the following structure represents a typical large Power Platform implementation project life cycle.

In this section, we will provide a detailed description of the typical activities and outputs in all of the sub-phases of such a project for both the customer and the implementation partner.

### **The preparation phase**

The **preparation** phase represents everything that happens before the implementation actually starts. Let's have a look at some of the elements of this phase.

#### **Identifying demand**

At the very beginning of any potential Power Platform implementation, an idea or **demand** for a new IT solution is identified in a company. The reasons for an identified demand can be very varied. It can be purely an idea that some business processes need to be automated, as it is no longer feasible to work with paper notebooks or Excel. It can also be the need to replace an outdated legacy IT system that is not flexible, cannot adapt to the accelerated changes in the business, or is simply running out of any feasible support scheme.

The drivers generating this demand are usually the business or IT department or both, and this driver is usually influencing all subsequent steps and the whole project preparation and execution process.

After the demand is identified, an internal discussion and evaluation usually starts and leads to a potential decision that the demand is legitimate. The usual next step is a feasibility evaluation to perform a first detailed analysis.

## **Studying feasibility**

A possible next step after the demand is recognized is a **feasibility** study to analyze the demand and evaluate the details. The feasibility study can be performed internally or by appointing an external consultancy. The purpose and outcome of the feasibility study should be to identify economic, technical, legal, operational, and scheduling feasibility.

The content of these feasibility areas is as follows.

### **Economic feasibility**

**Economic feasibility** evaluates the potential business and financial benefits, cost/benefit analysis, expected **Return Of Investment (ROI)**, financial projections, risk analysis, and finally, a high-level cost estimate.

### **Technical feasibility**

**Technical feasibility** evaluates the IT environment, the expertise of the company's staff to run the potential implementation, the possible technical solutions on the market, a make-or-buy decision, and a decision for an on-premises, cloud, or hybrid solution.

### **Legal feasibility**

**Legal feasibility** evaluates all possible and involved legal factors for the planned solution, including compliance requirements, data protection and processing requirements, and data residency requirements.

### **Operational feasibility**

**Operational feasibility** evaluates the impact of the solution on possible reorganizations, additional staffing, readiness, and other possible organizational changes.

### **Scheduling feasibility**

**Scheduling feasibility** evaluates the possible timeline for the implementation considering all constraints such as resource availability, business closures, and legal and financial reporting periods. The feasibility further analyzes the potential impact of project failure at a certain point in time or project extensions caused by external factors.

The primary outcome of the feasibility study should be a **go or no-go decision** with proper justification. In case of a go decision, the feasibility study should frame the next steps, specifically, the following:

- Budget allocation
- Internal or external project decision
- Internal resources allocation
- High-level project schedule
- High-level technology decision
- Vendor and implementation partner selection process

## **Specifying the budget**

With a go decision and high-level budget decision, there could be a further **budget specification** such as the budget framing, the possibility of extending the budget, sources of financing, cost allocations and distributions to impacted business units in the organization, and other detailed decisions.

## **Seeking approval**

The most important part of the preparation phase is the final **approval** from top management to start the procurement process of selecting and contracting the vendors and consultancies. After this step, the company starts communicating externally and involving external subjects in the selection and bidding process.

## **Issuing a Request for Information (RFI)**

After formal approval, the next step could be issuing an **RFI**. The purpose of the RFI is to gather comparable information from a broad selection of vendors or implementation partners about possible products and services they can offer to fulfill the requirements. Compared to the next step, the RFI requires fewer details from the vendors or implementation partners and is issued to more recipients. The RFI request can be more or less formalized, but for the purpose of easy evaluation, it is recommended to design the request with a strict formalized structure with clear and simple response options. Part of the RFI request could be the following:

- Basic commercial information about the vendor/implementation partner
- A brief requirements catalog or description accompanied by a dedicated questionnaire about the main capabilities of the products and services
- A basic schedule and pricing estimates

The evaluation of the RFI responses usually leads to a **shortlist** vendors or implementation partners that are invited to respond to a **Request For Proposal (RFP)**, **Request For Quotation (RFQ)**, or **Request For Tender (RFT)**.

## Issuing an RFP/RFQ/RFT

RFP, RFQ, and RFT are names for similar types of bidding process, where the customer requires much more detailed information from the recipients than during the RFI step. The customer usually requires the following information:

- Extended commercial information about the vendor/implementation partner
- Answering the detailed requirements catalog with functional and non-functional requirements for the planned solution
- High-level solution architecture
- High-level project plan and project schedule
- Basic information about the key potential project team members
- Detailed pricing information broken down into various pricing categories

Answering RFP/RFQ/RFT requests from potential customers is a complex and time-consuming process, and vendors and implementation partners usually try to standardize the process using various tools and templates. At the same time, they need to book appropriate time with the various experts required to support answering the request.

The customer sometimes requires from the partner to present the proposal or quotation in an onsite meeting or conference call presentation to the stakeholders for the chance to get to know the possible key project team members personally and ask detailed questions.

Part of this bidding process step could be a request to the potential vendors or implementation partners to prove their capabilities by implementing and presenting certain product or solution functionalities either as a simple **product or solution presentation** but also as a **Proof of Concept (POC)** or even a **hackathon**, as described in the next sections.

### Proof of concept

A POC is an extended presentation of the capabilities of the products and solutions. The customer defines certain specific functional and non-functional requirements and gives the potential vendors or implementation partners the opportunity to implement the required functionality and present the results to the stakeholders.

## **Hackathon**

A **hackathon** is sometimes used by customers instead of a POC to verify how the potential vendor or implementation partner is able to implement the required functionality within a very short timeframe, under pressure, and in a controlled environment. The customer invites an implementation team to their premises and presents the requirements shortly before the hackathon starts and gives the team an exactly defined timeframe to finish the solution development. The results are then presented to the stakeholders and are considered in the bidding process.

The RFP/RFQ/RFT response needs to have a clear structure and, usually, depending on the customer request, consists of these components:

- Questionnaire response
- **Statement Of Work (SOW)**
- Effort estimate and pricing
- Project schedule
- Additional required responses

Here are some additional details of the main parts of a response:

- The **questionnaire response** is usually a structured document (Word or Excel) with questions related to functional and non-functional requirements. It is expected from the partner to provide a response description, type, and level of coverage.
- The **SOW** is a legally binding document describing the services to be delivered within the project. Vendors and implementation partners usually use internally approved templates that are tailored to the specifics of the individual project.
- The **effort estimate and pricing** is a cost calculation performed by the partner expert team, usually consisting of a sales person, a solution architect, and a project manager, as described earlier in this chapter.
- The **project schedule** is created with respect to the effort estimate to provide an understanding of the project duration, phases, and so on.
- The customer may require various additional response material, such as a signed **Non-Disclosure Agreement (NDA)**, partner financial statements, and product or project references.

The RFP, RFQ, or RFT step should lead to the **selection** of a final vendor or implementation partner, and every subsequent step is carried out between the customer and the winner of the bidding process.

## Discovery

**Discovery** is an optional step that can occur either in the preparation phase or at the beginning of project execution and can be provided as a free of charge or already paid service by the implementation partner. The purpose of the discovery is to better understand the customer requirements and the vision of the required solution. The discovery is usually a short activity taking days or a few weeks of meetings and documentation analysis between the customer and the implementation partner experts. The output of the discovery can be more detailed scoping and improved effort and cost estimates.

## Negotiations

The **negotiations** step is the last step before a contract is signed and usually happens between the customer procurement team and the vendor or implementation partner sales team. The partner is already selected, and, in this step, the commercials, pricing, scheduling, and legal open questions are finalized, and the contract is prepared for signing.

It is recommended for this phase of the project preparation to perform a **soft booking** of the key resources needed to start the project execution since, after the contract signing, the customer is usually keen to start the project execution as soon as possible.

## Contract

The **contract** signing is the very last step of the project preparation. Immediately after the signing, the project execution starts.

## The project execution phase

**Project execution** is the contractual phase of the project, where the implementation partner in cooperation with the customer performs all the work to fulfill the project objectives. In this section, you will learn more details about each project execution phase.

The presented project phases are tailored to the *iterative project model*, which is most widely used in large Power Platform implementations.

## Project preparation

The first phase in the project execution is the **preparation of the project** before the actual work starts. Most of the activities in this phase are performed by the implementation partner. This phase usually takes several weeks and consists of the following main tasks:

- The final assignment of all partner project resources: If the possible project resources were soft booked during the contract negotiations, they must now be finally booked for the duration of their planned engagement.
- In cooperation with the customer, plan and book customer resources.
- If an onsite presence is required, prepare the project locations, rooms, workplaces, hardware, and software in case the customer's infrastructure is needed, internet access, and many other small project location-relevant preparations.
- Manage any travel requirements for booked onsite resources not situated close to the project location.
- Prepare resources and materials for the project initiation phase.
- Detail and finalize the project plan and other project-related documentation.

## Project initiation

The official start of the project is the **project initiation**. The main activity within this phase is, besides many preparation steps, the **kick-off meeting**. This phase already fully involves the customer, usually takes several weeks, and consists of the following main tasks:

- Preparation of the kick-off meeting
- Project management activities, such as workshop planning, governance and methodology alignment, communication planning, project reporting approach planning, and internal project team alignment meetings
- Setting up any necessary software, especially Azure DevOps, any Power Platform environments, on-premises software on the team members' PCs, and so on
- Conducting the kick-off meeting

After the kick-off meeting, the real daily project execution starts with the initial analysis.

## Initial analysis

Since the *iterative project* approach is less dynamic than the *agile* approach, there is a non-iterative **initial analysis** phase that's used to prepare the project for the iterative execution phase. The initial analysis usually takes several weeks and consists of the following typical tasks:

- Conduct a series of business-related workshops dedicated to workload categories between the implementation partner and customer project experts. The typical participants are project managers, the partner solution architect and business consultant(s), the customer business process owner(s), and other relevant roles.
- Conduct a series of technical workshops dedicated to technical architecture, infrastructure, security, integration, data migration, and other technical topics. The typical participants are project managers, the partner solution architect and technical consultant(s), the customer IT lead, enterprise architecture, and other relevant roles.
- Present the out-of-the-box capabilities of the Power Platform and Dynamics 365 components, identify and document the gaps against the requirements.
- Refine the requirements document and validate all requirements with the customer.
- Enter the requirements structure into the Azure DevOps environment.
- Finalize the content for each of the planned iterations for the iterative executing phase.
- Update the project plan.
- Create initial versions of the solution architecture, solution design, and other main project documents.

At the end of the initial analysis, there should be the following results as prerequisites for starting iterative execution:

- The business processes and business requirements are refined, validated, documented, and entered into Azure DevOps.
- The technical requirements are validated, documented, and entered into Azure DevOps.
- The content for all iterations is defined.
- The initial versions of all required project documents are finalized and accepted.

## Iterative execution

The **iterative execution** is the main and the longest part of the project. This part consists of a defined number of iterations in which the solution is subsequently developed and tested. Every iteration consists of four sub-phases: analysis, design, development, and testing.

The sub-phases overlap as described earlier in this chapter, so that the analysis and design of the next iteration is performed during the development and testing of the current iteration. The benefit of this approach is faster project progress and better utilization of specialized resources since they do not have repeated gaps in their activities. This approach, however, requires a high degree of coordination within the project team.

The typical activities and outcomes of the sub-phases are described in the next sections.

## **Iterative analysis**

The main purpose of the **iterative analysis** sub-phase is to fully focus on the requirements specified for the iteration and prepare every detail for design and development. Typical activities in iterative analysis are as follows:

- Analyze and categorize the requirements.
- Conduct workshops for every requirement category with the respective experts (business requirements, integration requirements, data migration requirements, security requirements, reporting requirements, and so forth) to fully document the requirements details.
- Decide on an implementation approach for the requirements.
- Proceed with the technical preparation for the subsequent sub-phases (prepare IT systems for integration development and data migration processes).
- Prepare test scripts.
- Update the project plan, Azure DevOps board, solution architecture, solution design, and other main project documents.

At the end of the iterative analysis sub-phase, there is clear and documented scope for the subsequent iterative design sub-phase.

## **Iterative design**

The main purpose of the **iterative design** sub-phase is to define how to implement all of the requirements analyzed and documented in the iterative analysis sub-phase. A typical activity in the iterative design is to conduct workshops for every requirement category with the respective experts (business requirements, integration requirements, data migration requirements, security requirements, reporting requirements, and others) to fully document the implementation design of the requirements.

The design specification can be anything, including the following:

- Specifications for Dataverse data model extensions
- Specifications for creating model-driven applications and updating the user interface of model-driven applications
- Specifications for creating automations with business rules, workflows, business process flows, and Power Automate

- Specifications for configuring business units, security roles, and field security profiles
- Specifications for developing **Power Apps Component Framework (PCF)** controls
- Specifications for developing custom Power Platform data connectors
- Specifications for creating automations with PlugIns and Custom Workflow Actions
- Specifications for creating canvas apps
- Specifications for creating Power BI datasets and reports
- Specifications for creating integration components using specified Azure services
- Specifications for creating data migration scripts
- Updating the project plan, Azure DevOps board, solution architecture, solution design, and other main project documents

At the end of the iterative design sub-phase, there is clear and detailed documentation for the subsequent iterative development sub-phase.

## **Iterative development**

The main purpose of the **iterative development** sub-phase is to perform the actual work on the systems as specified in the iterative analysis and design. Typical activities in iterative development are as follows:

- Perform customization and configuration of the Dataverse database, model-driven apps, canvas apps, Power Automate, Power BI, and so on.
- Perform custom development tasks in every part of the overall solution (Dataverse, PCF components, custom connectors, Azure, and on-premises).
- Develop data migration scripts.
- Develop test scripts.
- Perform unit testing of the developed artifacts.
- Perform application life cycle management activities (such as solution export and import, or committing the solution and custom development artifacts into repository).
- Update the project plan, Azure DevOps board, solution architecture, solution design, and other main project documents.

At the end of the iterative development sub-phase, a set of solution configurations, customizations, and custom development artifacts are prepared for the subsequent iterative testing sub-phase.

## **Iterative testing**

The main purpose of the **iterative testing** sub-phase is to perform all necessary testing of the artifacts developed in the iterative development sub-phase. While the iterative development sub-phase contains unit testing as part of the development process, iterative testing consists of a series of high-level testing activities. Moreover, the difference is that the iterative testing activities are performed in a test environment and not a development environment. Typical activities in iterative testing are as follows:

- Create new or update existing functional, system integration, and user acceptance test scripts.
- Manage test data in the test environment.
- Conduct the functional and system integration tests.
- Update user training material.
- Record test results in the test recording tool (Azure DevOps Test Plans).
- Update the project plan, Azure DevOps board, solution architecture, solution design, and other main project documents.

At the end of the iterative testing sub-phase, there is a set of documented testing outcomes that can be used to correct all defects and modify all developed solution capabilities.

## **Final testing**

The **final testing** phase happens after all iterations are finished and the Power Platform solution is considered ready. The purpose of the final testing is to perform all end-to-end tests of the solution to verify that the solution fulfills customer requirements and to prepare the solution for deployment. This phase usually takes several weeks, and the main activity is the official **User Acceptance Testing (UAT)**. The testing activities are performed in a dedicated UAT environment. Typical activities in the final testing phase are as follows:

- Finalize the testing strategy and UAT test scripts.
- Prepare a Power Platform UAT testing environment, load the final solution, and test data.
- Perform key user/train-the-trainer training.
- Perform the UAT.
- Perform the final data migration tests.
- Finalize training materials.

- Prepare a Power Platform production environment.
- Update the project plan, Azure DevOps board, solution architecture, solution design, and other main project documents.
- A decision for deployment and go live is made.

At the end of the final testing phase, everything is prepared for the last project execution phase—solution deployment.

## Solution deployment

The **solution deployment** phase is the last project phase before going live with the solution. Typical activities in the solution deployment phase are as follows:

- The final tested and corrected solution is deployed on the production environment.
- Final configuration of all solution components is performed in the production environment.
- The final migration of production data to the production Power Platform environment is performed.
- Smoke testing on the production environment is performed.
- Go live with the new solution is confirmed.

With the conclusion of the solution deployment phase, the project execution is finished, and the customer starts using the new solution. It is not uncommon for the customer's operational team to be involved in these activities to learn and take ownership. In some organizations, there are access policies in place prohibiting external vendors' access to production environments. In this case, those activities must be performed by the customer's staff. The role of the implementation partner is reduced, and the management and maintenance of the new solution is transferred to the customer's support department. The implementation partner, however, usually provides some limited continued support as described in the next section.

## The operation phase

The **operation** phase is the phase during which a finally deployed IT solution is productively used by the customer. The main activity during this phase is to ensure that the IT solution is working properly and delivering the expected business value. At the beginning of this phase, the implementation partner provides certain contractually defined services to help the customer to fully establish their own support.

## **Support transition**

If contractually specified, there could be a certain period of time after the solution deployment during which the implementation partner provides **support transition services**. The duration of this phase is usually several weeks. Typical activities in the support transition phase are as follows:

- Training activities for the customer's support organization
- Joint customer support for the deployed solution (job shadowing)
- Preparations for possible long-term support, according to a separate support contract
- Handing off customer support from the implementation partner to the customer's own support organization

After the support transition phase, the customer's support organization is able to take full responsibility for providing customer support for the deployed solution.

## **Operation**

The standard operation phase can be very long and will accompany the use of the Power Platform solution in production until the solution reaches its end of life. The most important activities during this phase are maintenance and support of the solution, which are described in the next sections.

### **Maintenance**

A Power Platform solution is a cloud-based solution with possibly few on-premise components and, as such, does not need extensive maintenance. Here are the typical maintenance activities that must be covered by the customer's support organization:

- **Maintaining Microsoft product updates**, especially installing updates for the first-party application (Microsoft Dynamics 365), which are not installed automatically
- **Manual backups** of the Power Platform environments if any larger changes on the environments are planned
- **Testing** the running Power Platform solution on a preview of a **new major platform update**
- Maintenance of any **on-premises** solution components

Extending the existing solution with new capabilities is not considered maintenance and usually requires a new project to be set up.

## Support

The **support** of a Power Platform solution in production is not different from the support of any IT system in an organization and it is not in the scope of this book to dive deep into how standard IT support works within an organization. The purpose of the support is to capture end user issues and resolve them in a structured and timely way. The most typical structure of a supporting organization for a Power Platform solution consists of three levels:

- **First-level support:** This is usually provided by the customer's own helpdesk organization. The first level should be able to resolve simple repeated questions and issues coming from end users.
- **Second-level support** is the first escalation layer for issues not covered by the first level. The scope of the second level is usually support regarding issues with the solution delivered by the implementation partner. That's why it is sometimes covered by a separate support contract with the implementation partner to forward some second-level issues to them for resolution.
- **Third-level support** is the last escalation layer. The scope of the third level is usually support for product-related issues. That's why the third level is usually provided by Microsoft either directly from the level of the customer or indirectly, when the implementation partner is assessing the issues and requesting support from Microsoft on behalf of the customer.

## Decommission

Every IT solution will reach its end of life at some point. If a Power Platform solution needs to be decommissioned and replaced with another solution, there is usually one single concern: the data managed in the solution. A Power Platform solution based on Dataverse stores its data in a cloud database. Microsoft provides the possibility to hand over the database to the customer when the Power Platform subscription ends. This gives the opportunity to perform a well-structured data migration from the Dataverse database to the successor solution.

## Contoso Inc. starting the implementation project

Contoso Inc. has analyzed the specifics of a Power Platform implementation project and decided to go ahead with the project preparations with the goal of selecting the best suitable implementation partner and starting the project. They have made a series of preparation activities and have made some important decisions.

## Bidding process

Since Contoso Inc. decided upfront to implement a Power Platform solution (i.e. the vendor decision is taken), they initiated a bidding process for an implementation partner only. After a structured process consisting of an RFI, RFP, POC, and final negotiation, the consulting company, **Proseware Inc.**, was contracted to perform the Power Platform implementation.

### Important note



**Proseware Inc.** is a fictitious large consultancy holding the Solutions Partner designation badges for the specialization Business Applications, Modern Work, and Data & AI. The company has a worldwide presence with subsidiaries in North America, Europe, and Asia.

## Project setup and methodology

A joint decision between Contoso Inc. and Proseware Inc. was made to structure the implementation into a Power Platform program with three separate projects:

- **A Power Platform business solution project** managed using the **iterative** project implementation model: This project will cover the implementation of the business processes in all Microsoft Dynamics 365 CRM modules as decided at the beginning. Part of the project will be implementing any possible custom model-driven applications and other components of the Power Platform covering business requirements. The decision for an iterative model was chosen to ensure that business value from the solution will be available soon as well as to ensure deep involvement of Contoso's key users in the solution development.
- **A Power Platform integration solution project** managed using the **waterfall** project implementation model: This project will cover the integration of the Power Platform solution into the existing Contoso Inc. IT ecosystem, including security integration, integration with a selected number of existing legacy IT systems, and any other required technical integration. The waterfall model was considered the most suitable, since the implementation of the integration solution will need longer preparation times and organization and management of many external stakeholders.

- A Power Platform data migration project managed using the **waterfall** project implementation model: This project will cover the migration of various data types from a lot of very different legacy IT systems that are planned to be replaced by the new Power Platform solution. The decision for the waterfall model for data migration was taken for similar reasons as for the integration project.

## Project plan, tools, and documentation

Contoso Inc. and Proseware Inc. made the following decisions:

- **Project plan:** Part of the contract with Proseware Inc. is a project plan for the implementation of the overall Power Platform solution for a total duration of **18 months**.
- **Project management tools:** It was decided to use *Azure DevOps* as the overall project management tool for the whole program including all three projects. An appropriate setup of the tool will be prepared.
- **Project documentation:** It was decided to use the following main project documents: program and project plan, requirements document, solution architecture document, functional design document, technical design document, test strategy document, test cases document, and training documentation.

Even though the program will run in three separate projects, there will be consolidated project documentation across all three projects.

## Project setup

Contoso Inc. decided to follow the established best practices for large IT projects and set up a mirroring project organization together with Proseware Inc. that has program managers, project managers, and all other roles as illustrated in the following project organization diagram:

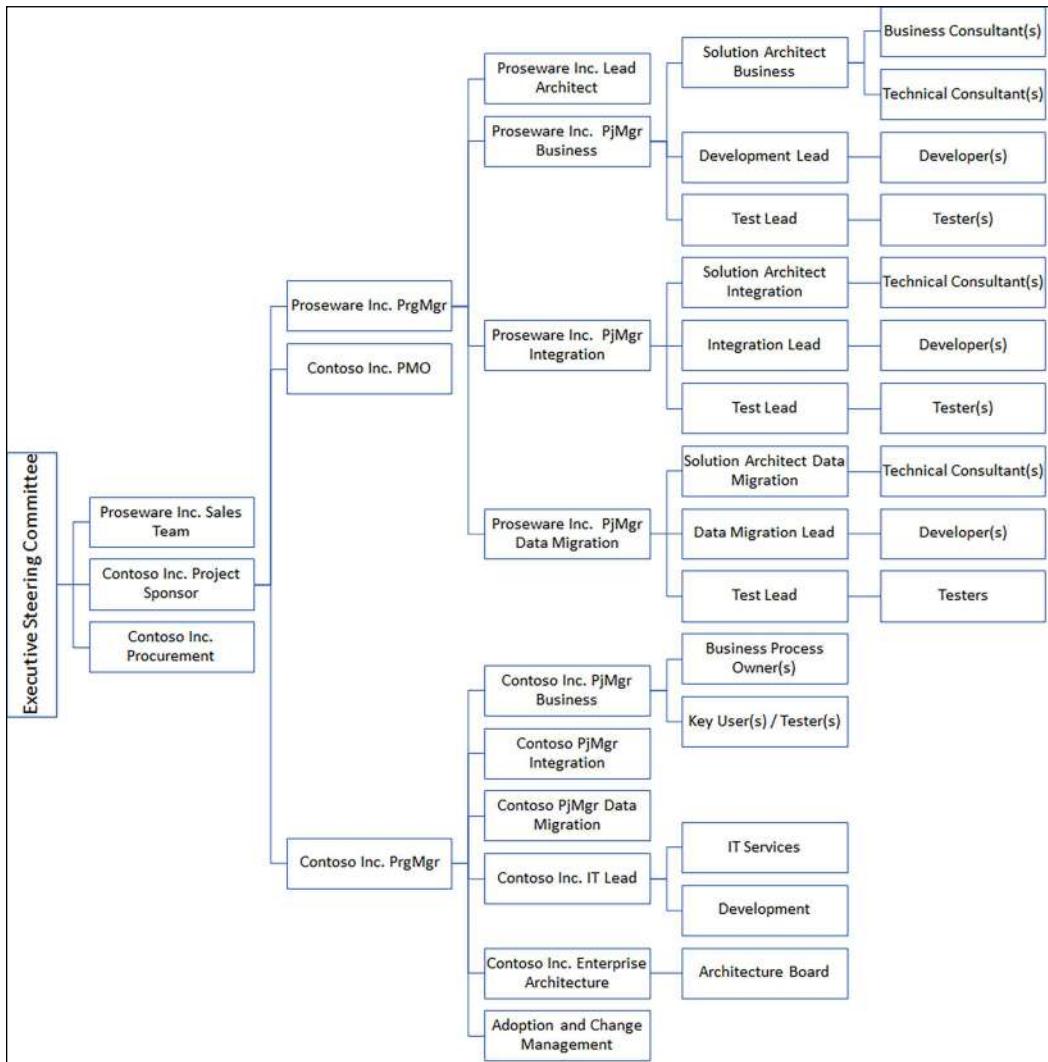


Figure 6.7: Contoso Inc. Power Platform project organization

The roles and responsibilities in the project setup will be assigned according to the following principles:

- The Contoso Inc. and Proseware Inc. **program managers (PrgMgrs)** will work closely together to guide the program execution in the desired direction. They will report to the project sponsor.
- Both parties will have three **Project Manager (PjMgrs)**, working closely together in the assigned areas of developing the business solution, developing the integration, and preparing the data migration. They will be responsible for the daily project management tasks and report to the program managers. The project will use the existing Contoso Inc. **PMO** to support the program.
- Proseware Inc. will assign a lead architect, responsible for the overall architecture of the solution. The lead architect will provide guidance for the solution architects and be part of the Contoso Inc. architecture board.
- Proseware Inc. will establish a project organization for the three projects containing all of the necessary roles for the respective project. The projects will be managed by the assigned project managers and responsibility for the solution will be held by the respective solution architects, who will also be temporary members of the Contoso Inc. architecture board.
- Contoso Inc. will establish a project organization for the three projects containing all of the necessary roles for the respective project. The business owners will be the main subject matter experts for developing the business processes. The key users will be involved in the development of the solution as testers from the very beginning. The IT lead with their organization will provide all the necessary IT and development support for the project. The adoption and change management department will work closely with the project team to prepare all the prerequisites for successful adoption of the new Power Platform solution.

At this point, Contoso Inc. is ready to start the Power Platform project execution. The implementation partner is contracted, the project setup, and methodology are agreed and Contoso Inc. together with Proseware Inc. can look ahead to the first practical steps in the project.

## Summary

In this chapter, you have learned a lot of practical details about how to structure a Power Platform implementation project, what the best project methodologies are, what documentation usually needs to be created, what the typical project organization structure is, and how a complex Power Platform project life cycle usually looks. With this knowledge, you should be able to set up a Power Platform project properly and make sure the implemented solution will fulfill the requirements.

In the next chapter, you will learn about all aspects of security and compliance when building a Power Platform solution.



# 7

## Microsoft Power Platform Security

In this chapter, we will fully focus on all the aspects of **Microsoft Power Platform security**. Understanding the security possibilities of the Microsoft cloud services, more specifically Power Platform, is a key requirement to be able to design and implement a mature security model.

We are going to cover the following main topics:

- Power Platform security overview
- Power Platform authentication
- Power Platform authorization
- Power Platform compliance, privacy, and data protection
- Security best practices
- Contoso Inc. security architecture

### **Contoso Inc. designing Power Platform solution security**

After selecting and contracting Proseware Inc. and kicking off the solution implementation project, Contoso Inc. starts architecting and designing the future Power Platform solution together with its implementation partner. Since Contoso Inc., as a global corporation with a high market reputation, takes IT security very seriously, the first step in project analysis and design will be to establish a **mature security model**. To achieve this, Contoso Inc. will need to fully understand all the security capabilities and adopt the necessary best practices.

## Getting an overview of IT security

IT security is a very complex area with the ultimate goal of protecting IT systems and IT networks from damage and disruption, and protecting data and other assets from theft and misuse. Security has grown from a small and not-so-important aspect in the past to one of the key pillars in today's business world, and its importance is even higher with the rise of cloud computing. IT security can be categorized into the following main areas:

- **Security:** Dealing with safeguarding hardware and software from any damage and unauthorized access.
- **Privacy and data protection:** Dealing with protecting the data that's managed in IT systems from theft and misuse.
- **Compliance:** Dealing with meeting legal IT-related obligations of the government and other authorities.

In this chapter, we will describe all three areas, with a special focus on the Microsoft Power Platform security aspects.

## Authentication versus authorization

Granting access to any IT system and, in particular, to a cloud service, always consists of two main steps:

- **Authentication:** This is the process of verifying the identity of the user or service principal that is trying to access a cloud service resource. The authentication process only verifies the identity, but without the subsequent authorization, it does not grant any privileges within the service. Within the Microsoft cloud environment, the authentication process is unified, regardless of the cloud service type, and always relies on **Azure Active Directory (AAD)**.
- **Authorization:** This is the process of granting the user permissions to use the service, access data, trigger business processes, and similar capabilities within a particular cloud service or solution. Within the Microsoft cloud environment, the authorization process is mainly implemented inside the cloud service itself, so that the authorization approach can differ from one service to another.

Next, we will have a look at the fundamentals of authentication and authorization in the Microsoft cloud ecosystem.

## Microsoft cloud authentication and authorization fundamentals

Before any user can access any Microsoft cloud service, including the Power Platform services, there is a standard procedure that must be followed in order to onboard the user so they can access the service. This procedure consists of three fundamental steps:

- **Provision user identity:** This establishes an identity for a new user in the cloud environment.
- **Assign licenses:** This gives the new or existing user commercial permission and technical access to use a certain cloud service.
- **Grant authorization:** This configures access rights for the user within the respective cloud service.

The **authentication** and **authorization** concepts for Microsoft cloud services are illustrated in the following diagram:

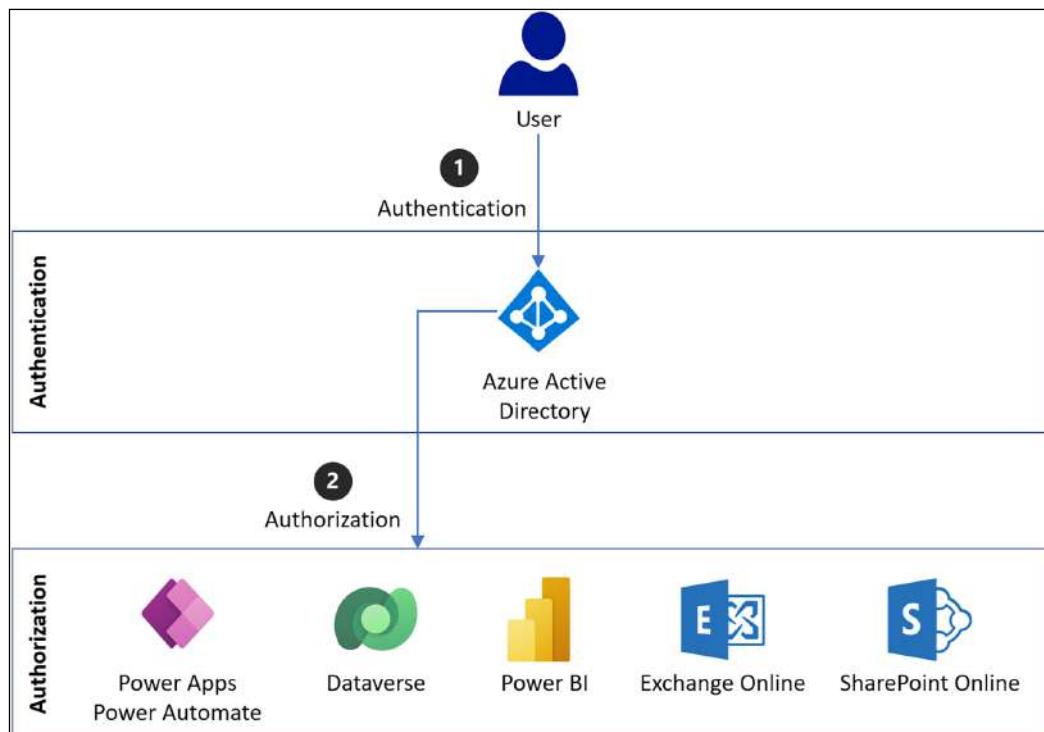


Figure 7.1: Microsoft cloud authentication and authorization

As shown in the preceding diagram, every user is first authenticated in the AAD to verify their identity. Authorization – in other words, which data and capabilities can the user use within the cloud service – is managed within the cloud service itself. In the previous diagram, authorization in different cloud services is managed by the following concepts:

- **Power Apps and Power Automate flows:** Authorization within apps and flows in a Power Platform environment without Microsoft Dataverse is managed using built-in, non-modifiable environment roles.
- **Dataverse-based applications:** Authorization within Dataverse-based applications is managed by a very detailed authorization concept consisting of business units, security roles, hierarchy security, teams, and column-level security.
- **Power BI:** Authorization within Power BI is managed by *granting access* to parts of Power BI, such as the workspaces for individual users or AAD groups. In addition, Power BI provides us with the ability to define row-level security to restrict access to records in the Power BI datasets based on defined rules.
- **Exchange Online:** Authorization within Exchange Online is managed using admin and user role groups and roles with configured permissions (**role-based access control (RBAC)**).
- **SharePoint Online:** Authorization within SharePoint Online is managed using a concept of permission levels and permissions, which can be assigned to individual users or AAD groups individually on each SharePoint site collection.

For other cloud services within the Microsoft cloud ecosystem, there might be different authorization concepts.

In the next few sections, you will learn more about the options that are used for the three basic steps of enabling a new Microsoft cloud user, as mentioned earlier in this section.

## Provisioning user identity

The Microsoft cloud with its capabilities (Microsoft Azure, Microsoft 365, and Microsoft Power Platform) uses AAD as its centralized identity provider. Provisioning a new user in the AAD can be done in a variety of ways, and there are also several types of identities in the AAD. You will learn more details about the identity types and provisioning possibilities in the next few sections of this chapter.

## Assigning licenses

The second step in providing user access to a cloud service is assigning them a license for the respective service. Depending on the service type, assigning the license can trigger certain background processes to enable the user in that service. Assigning a license can be done manually in the Microsoft 365 or Azure administration portals, using *PowerShell* or the *Microsoft Graph API*.

## Granting authorization

The last step for enabling a user is to grant them authorizations directly in the respective cloud service. This process is very different for the various cloud services within Power Platform, going from the simplest authorization for canvas apps or Power Automate flows, through the medium complexity implemented in Power BI, to a very sophisticated and detailed authorization concept in Dataverse and model-driven applications.

In the following section, you will learn about authentication within Microsoft cloud services.

## Understanding authentication

In this section, we will describe the details of authentication in the Microsoft cloud ecosystem, which is generally valid for all Microsoft cloud services, including Power Platform. First, we will cover the authentication of internal organizational users and then look at the authentication capabilities for guest and external users.

## Identity and authentication solutions for internal users

Every internal user must be a member of the tenant's AAD. Organizations can decide to use cloud identities only, where user management can be performed solely within AAD. Large organizations with complex internal IT infrastructure, however, require a certain level of integration for their existing on-premises Active Directory structures with the cloud. There are many reasons for this, but mainly, the requirements are to keep a consolidated process of provisioning user identities and to keep full control of security within its own boundaries. There are several possibilities regarding how to integrate AAD with an organization's on-premises Active Directory in order to achieve a hybrid identity. In the next section, you will learn about the possible identity scenarios you may encounter.

## Cloud identity approach

Using cloud identity is the most simple, non-hybrid approach and is available out of the box for every Microsoft cloud service. This approach is illustrated in the following diagram:

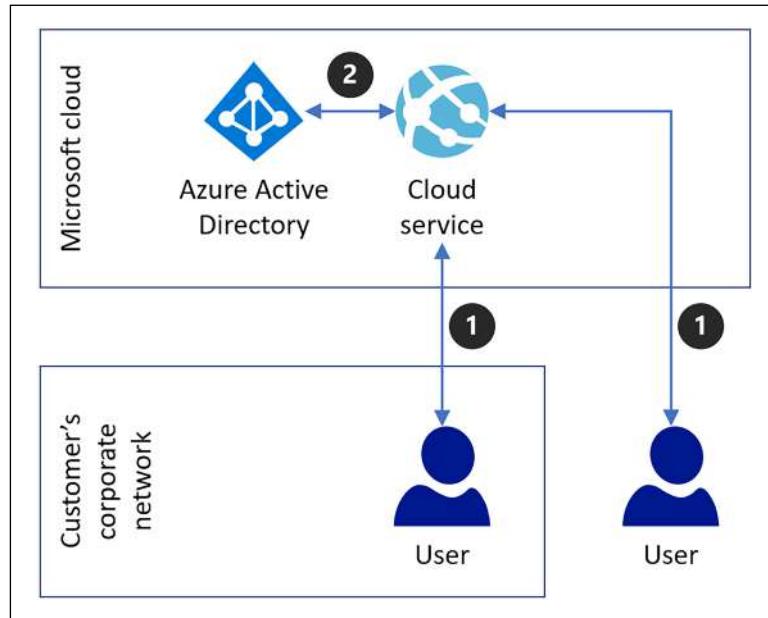


Figure 7.2: Authentication with cloud identity

As we can see, authentication happens only in the cloud, and is implemented as a simple two-step process:

1. The user requests access to a cloud service.
2. The cloud service enforces a login process and verifies the identity within **Azure Active Directory**. When the identity is validated, access to the cloud service is granted.

This authentication type in the basic configuration does not enforce any constraints regarding the user's location, so a user can authenticate from within the own organization's corporate network, as well as from any public internet connection. Depending on the subscription levels of Microsoft 365 and Azure, the cloud identity approach offers the following additional capabilities:

- Conditional access
- **Multi-factor authentication (MFA)**
- Self-service password reset

You will learn more about these capabilities later in this chapter.

## Password hash synchronization approach

Password hash synchronization is the simplest implementation of a hybrid identity. This approach is illustrated in the following diagram:

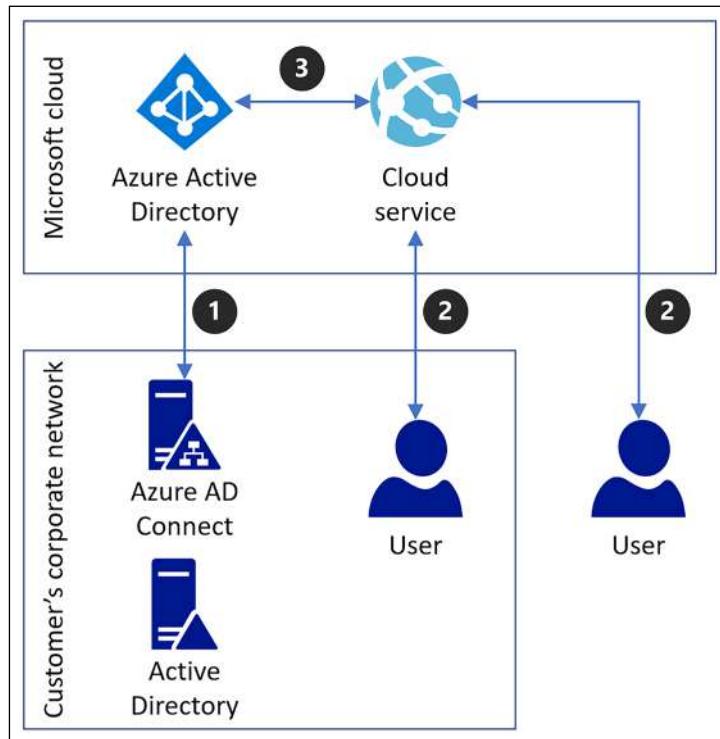


Figure 7.3: Authentication with password hash synchronization

As we can see, authentication still happens in the cloud and is implemented with the following steps:

1. The organization implements the **Azure AD Connect** component in their own IT infrastructure. This component must be configured to synchronize the user accounts and password hashes to **Azure Active Directory**.
2. The user requests access to the cloud service.
3. The cloud service enforces a login process and verifies the identity within **Azure Active Directory**. When the identity is validated, access to the cloud service is granted.

The benefit of this approach is that users can use **the same credentials** for the cloud service that they use for their on-premises IT solutions. When deciding to use this approach, the following must be considered:

- An on-premises infrastructure is needed to install and maintain the Azure AD Connect component. For failover safety, at least two instances of the service should be deployed as a cold or hot standby solution.
- The on-premises password, as a highly secured identity artifact, is synchronized in the cloud. Even though only a hash and never the clear-text content of the password is synchronized, some organizations might see this as a security risk and might not accept this approach.
- The approach supports a single sign-on experience using a capability called **seamless single sign-on**, which must be configured and enabled separately.
- The approach supports the same security capabilities as pure cloud identity since authentication happens in the cloud in the same way.

You will learn more about seamless single sign-on later in this chapter.

## **Pass-through authentication approach**

A more advanced hybrid authentication approach is called **pass-through authentication**, where the user's password is verified in the on-premises Active Directory. This approach is illustrated in the following diagram:

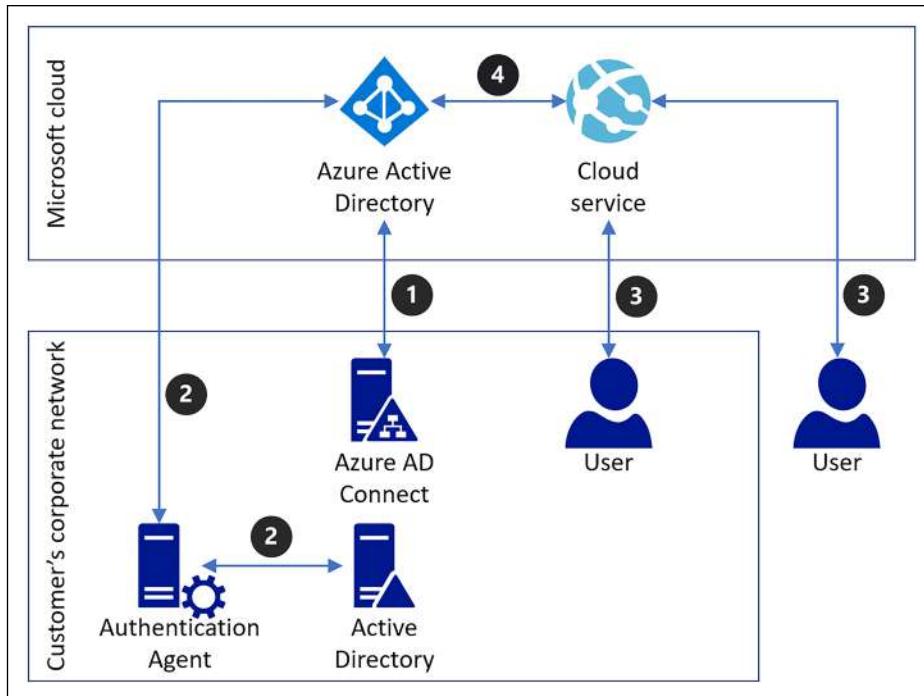


Figure 7.4: Authentication with pass-through password verification

As we can see, authentication is a two-step hybrid process. The implementation consists of the following steps:

1. **Azure AD Connect** must be configured to only synchronize the user accounts without passwords to **Azure Active Directory**.
2. The organization must also implement the **Authentication Agent** service to verify the passwords in the Active Directory.
3. The user requests access to the cloud service.
4. The cloud service enforces a login process, but password verification happens in the on-premises Active Directory via the Authentication Agent.

This approach is beneficial because users can use the same credentials for the cloud service that they use for their on-premises IT solutions. Furthermore, any on-premises account policy is enforced since every login attempt is verified with the on-premises Active Directory. When deciding whether you wish to use this approach, the following must be considered:

- An on-premises infrastructure is needed to install and maintain the Azure AD Connect component, along with the Authentication Agent. For failover safety, at least two instances of both services should be deployed.
- This approach also supports single sign-on using *seamless single sign-on*.
- This approach supports the same security capabilities as the *password hash synchronization* approach.

## Federation approach

The most advanced hybrid authentication approach is federation, where the user's credentials are completely authenticated in the on-premises Active Directory. This approach is illustrated in the following diagram:

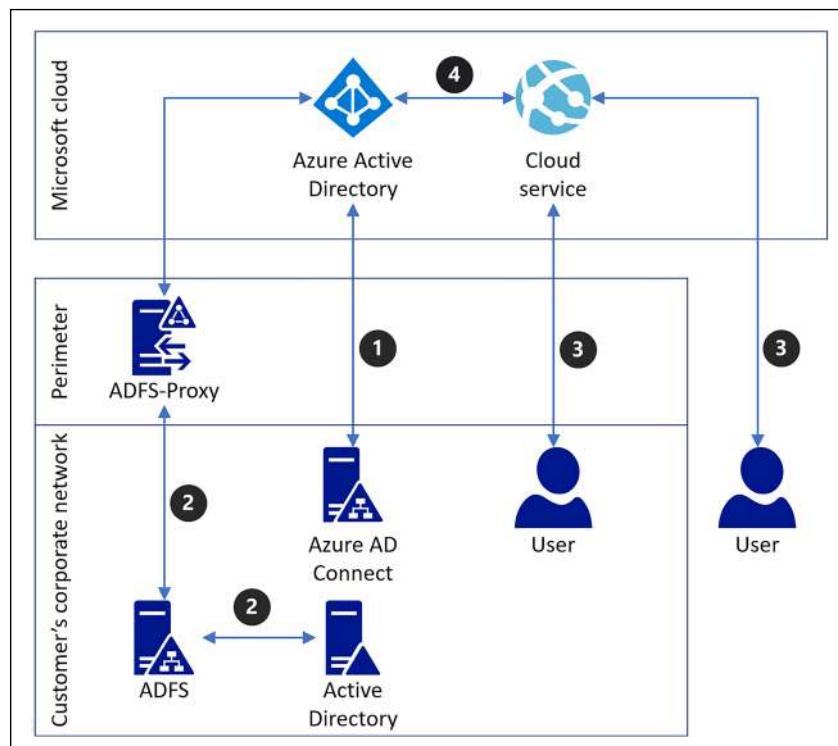


Figure 7.5: Authentication with Active Directory Federation Services (ADFS)

As we can see, the implementation consists of the following steps:

1. **Azure AD Connect** is configured the same way as in the previous approach.
2. The organization must, in addition, implement the **ADSF** to perform full authentication in the Active Directory.
3. The user requests access to the cloud service.
4. The cloud service recognizes the federation setup and forwards the authentication process entirely to the on-premises Active Directory, which verifies the identity and password. Next, the authentication result (authentication token) is sent back to **Azure Active Directory**, which validates the token and issues an access token that can be used to access the cloud service.

This advanced approach benefits us because the authentication stays completely under the control of the on-premises IT infrastructure and several advanced authentication scenarios can be implemented. When deciding on using this approach, the following must be considered:

- It is necessary to install and maintain the Azure AD Connect component, along with the ADFS solution. For failover safety, at least two instances of all the involved services should be deployed.
- The approach supports various non-standard requirements, such as an on-premises MFA solution with smart card- or certificate-based authentication, or integration with certain third-party authentication providers. It is also possible to integrate with multiple Active Directory forests. Those are all usual requirements for large multinational organizations.
- A single sign-on is available for all domain-joined devices.
- The approach also supports detailed security monitoring with Azure AD Connect Health.
- The ADFS components offer a wide variety of advanced conditional access settings using the ADFS claim rules.

You will learn more about ADFS claim rules later in this chapter.

## Conclusion

Selecting the right identity and authentication solution is a key decision for any organization planning to move to Microsoft cloud services. The selected solution will influence all the existing and future cloud services going forward. The most frequently used approaches are as follows:

- **Cloud identity:** This approach is used by smaller organizations, start-ups, or organizations fully committed to the cloud that do not see any added value to sticking with on-premises identities or non-Microsoft authentication solutions. The implementation is easy, and the security features are rich enough to fulfill legitimate security requirements.

- **Federation:** This approach is used by large, traditional multinational organizations with very high security-related requirements and with heavy investment in their own security infrastructure. The implementation needs more effort and requires certain additional on-premises infrastructure. When implemented correctly, it is an ideal enhancement for on-premises IT and the first step of the journey into the cloud.

There are certain additional authentication features that are important for a Power Platform solution. We will discuss these in more detail in the following sections.

## Authentication features for internal users

AAD and ADFS offer certain additional authentication features. These features are especially useful for Power Platform solutions. We'll learn more details about these features in the following sections.

### Conditional access

Conditional access is one of the most powerful AAD security features and is used to increase security and enforce policies. This feature makes it possible to observe signals coming from the user's devices, make decisions based on predefined rules, and enforce policies.

These signals can be any of the following:

- The user type or the membership of a user in a group
- The user's device type
- The application the user is trying to use
- The current location of a user
- Signals coming from other Azure security solutions, such as **Azure Identity Protection**

The decision that's made by the conditional access component can be either to grant access, block access, or require additional actions, such as requiring MFA.



#### Important note

The conditional access feature requires an Azure Premium subscription.

For Power Platform solutions, the most widely used scenario for conditional access is to restrict access to applications based on locations, for example:

- Restrict access to be able to *log in only within the corporate network*

- Restrict access to be able to *log in only from certain countries*

#### Important note



Conditional access for model-driven applications, including Dynamics 365, is enforced only during login user authentication. If the logged-in user breaks the policy during the live session – for example, by leaving the corporate network with their device – the policy will be enforced only after the current session times out and the next authentication request occurs.

## ADFS claim rules

ADFS is based on tokens containing claims to implement authentication scenarios. The solution provides a feature that can be used to implement business logic over the flow of tokens, as well as claims, by using the *ADFS claim rules*. The claim rules can implement various business processes, but for the purpose of Power Platform solutions, claim rules can be used specifically for the following:

- Location-based access restrictions
- Enforcing authentication security policies

As we can see, the possibilities for ADFS claim rules are similar to those for Azure conditional access.

## Multi-factor authentication

MFA is an extension of the standard authentication process, where the security principal provides one single security method for authentication, usually a password. MFA requires at least two of the following methods:

- **Something you know**, which is usually a password or another secret.
- **Something you own**, which is usually a physical device providing the authentication. It can be a hardware token generator, a mobile phone with authentication software, or a phone call or SMS for verification.
- **Something you are**, which is biometric information such as your face, iris, or fingerprint recognition.

MFA is fully supported for Power Platform-based solutions and can be implemented either within AAD or using a *third-party MFA solution*, if the federation identity approach is implemented.



#### Important note

Advanced MFA features require an Azure Premium subscription. The Azure Premium (P1 or P2) subscription is part of several Microsoft 365 subscriptions.

## Single sign-on options

For password hash synchronization and pass-through authentication, there is an option you can use to configure an AAD feature called **seamless single sign-on**. This feature works on all devices connected to the organization's on-premises Active Directory domain and avoids an additional sign-in dialog for logged-in users.

For the federation scenario, the authentication happens in the on-premises Active Directory, therefore the seamless single sign-on does not apply. ADFS, however, supports single sign-on using cookies or application-specific settings.



#### Important note

For federation scenarios that are integrating AAD with multiple independent on-premises Active Directory forests, there is always a selection dialog that appears so that the user can select the identity provider they wish to authenticate with.

Next, we will discuss a specific **Software as a Service (SaaS)**-related security feature called **cross-tenant restrictions**.

## Cross-tenant inbound and outbound restrictions

Certain security-sensitive organizations carefully manage access to public cloud content to enforce high IT security standards. This is traditionally done using IP or DNS name-based restrictions and filtering. But in the world of **SaaS** solutions, which are operated on shared infrastructure and often use standard public and shared URLs, this approach does not work.

To make it possible to enable access to SaaS solutions but restrict it for specified tenants only, Microsoft Azure offers a capability called cross-tenant restrictions. By enabling this capability, the organization can specify which tenants within a shared SaaS solution the users are allowed to use and vice versa, as well as which tenants are blocked. This capability can be used to restrict access in both directions separately:

- **Outbound restriction:** This restriction can be configured to block access to all SaaS cloud services in the specified tenant, or only access from Power Apps and Power Automate.

- **Inbound restriction:** This restriction can be configured to block tenant access for Power Apps and Power Automate.

#### Important note



In the Power Platform ecosystem, there are certain cloud services that use unique, customer-specific URLs, such as Dataverse-based (or Dynamics 365) applications. Using cross-tenant restrictions is useful only for services with shared, customer-independent URLs. With cross-tenant restrictions in place, it is still possible to configure cross-tenant access for selected services (selected Dynamics 365 environments, for example).

So far, we have discussed the authentication options for interactive user accounts. In the following section, we direct our attention to the authentication of service accounts used for API-based communication.

## Service authentication for internal users

Authentication is equally important when building any external solution components that will be connecting and communicating with the various Power Platform services via an interface. In this section, you will learn about the specifics of authentication against the Power Platform APIs.

### Dataverse authentication

Every external application that should connect to a Dataverse-based solution needs to be authenticated. The authentication type depends on the Dataverse API endpoint the application is using:

- For the older **SOAP-based endpoint**, there could be either Office 365 or OAuth authentication used.
- When using the modern **Web API-based endpoint**, the only authentication type available is OAuth.
- The latest **Tabular Data Stream (TDS) endpoint** uses an *AAD* username/password authentication via port 5558 to enable read-only access to Dataverse.

In the next few sections, you will learn more about Dataverse service authentication.

## Office 365 authentication

Office 365 authentication is simple but less secure, since it uses a username/password authentication type. A simple example of a *Who am I* request using Office 365 authentication is illustrated in the following code snippet:

```
using System;
using Microsoft.Crm.Sdk.Messages;
using Microsoft.Xrm.Tooling.Connector;

static void Main(string[] args)
{
    string url = "https://contoso.crm.dynamics.com";
    string username = "user@contoso.onmicrosoft.com";
    string password = "Pass@word1";
    string conn = $"Url = {url}; AuthType = Office365; UserName = {username}; Password = {password};";
    using (var svc = new CrmServiceClient(conn))
    {
        WhoAmIRequest request = new WhoAmIRequest();
        WhoAmIResponse response = (WhoAmIResponse)svc.Execute(request);
        Console.WriteLine("Your userid = {0}", response.UserId);
    }
}
```

As you can see, only the `url` of the Power Platform Dataverse environment, a `username`, and a `password`, are used to authenticate.

## OAuth authentication

OAuth authentication is mandatory when using the Web API endpoint and is optional for the SOAP-based endpoint. OAuth is more secure, since it requires every application to be first registered in the AAD of the tenant hosting the Power Platform environments.

### Important note



For details about the app registration in Azure Active Directory for OAuth authentication, please refer to the product documentation: <https://learn.microsoft.com/en-us/power-apps/developer/data-platform/walkthrough-register-app-azure-active-directory>

There are two OAuth authentication types:

- OAuth using a standard user account
- OAuth using an application user account

The first step in enabling OAuth authentication is to register the external application in the AAD. After the application has been registered, AAD provides a unique application ID (client ID) that is used during the authentication process to obtain an access token. In addition, the registered external application must be granted access privileges to Dynamics 365.

A simple example of a *Who am I* request using OAuth authentication with a standard user is illustrated in the following code snippet:

```
...
    string url = "https://contoso.crm.dynamics.com";
    string clientId = "51f81489-12ee-4a9e-aaaae-a2591f45987d";
    string username = "user@contoso.onmicrosoft.com";
    string password = "Pass@word1";
    AuthenticationContext authContext = new
    AuthenticationContext("https://login.microsoftonline.com/common", false);
    UserCredential credential = new UserCredential(username, password);
    AuthenticationResult result = authContext.AcquireToken(url, clientId,
    credential);
...
    string accessToken = result.AccessToken;
    using (HttpClient client = new HttpClient())
    {
        client.BaseAddress = new Uri(url);
        HttpRequestMessage request = new HttpRequestMessage(HttpMethod.
Get, "/api/data/v9.1/WhoAmI");
        request.Headers.Authorization = new
        AuthenticationHeaderValue("Bearer", accessToken);
        HttpResponseMessage response = client.SendAsync(request).Result;
...
}
```

As you can see, the authentication procedure is more complex as it is using the application ID (`clientId`) to request the access token from AAD. In the next step, the actual Dataverse transaction (the *Who am I* request) is executed with the access token in the header of the request.

This approach is more secure compared to Office 365 authentication since the application must be registered in the AAD first, and the authentication process must always use the unique application ID (client ID). However, for requesting the access token, the standard user credentials are used, which might be still considered less secure.

The most secure version of authentication is OAuth with the use of an application user account, not a standard user one. The difference is that, for an application user, no user credentials are provided in the authentication process, but rather the application ID (clientId) and a *client secret*, which is a unique key, generated during app registration in AAD.

In the following code snippet, the difference between standard user authentication and application user authentication is illustrated (the rest of this example is identical to the previous example):

```
string url = "https://contoso.crm.dynamics.com";
string clientId = "51f81489-12ee-4a9e-aaae-a2591f45987d";
string clientsecret = "<yourclientsecret>";
string tenantid = "<yourtenantid>";

AuthenticationContext authContext = new AuthenticationContext("https://
login.microsoftonline.com/common" + tenantid, false);
ClientCredential credential = new ClientCredential(clientId,
clientsecret);
AuthenticationResult result = authContext.AcquireToken(url, credential);
```

As you can see, the authentication process is using the URL, clientId, clientsecret, and the ID of the AAD tenant to perform the authentication process.

Another way to increase the security level is to use a certificate instead of the client secret. In this case, an X.509 certificate needs to be uploaded while registering the app in AAD, and the authentication code must include the certificate thumbprint rather than the client secret.

#### Important note



For the Web API endpoint, there is no specific Dataverse assembly provided by Microsoft. The whole communication uses the standard HTTP client library.

In the next section, we will discuss the specifics of authentication within Power BI.

## Power BI authentication

Connecting to the Power BI API requires authentication in a similar way that the Dataverse does, since the Power BI API is REST-based and the authentication type is OAuth. The first step in enabling OAuth authentication is identical to Dataverse – register your external application with AAD. This step can be performed either directly in the Azure administration portal or by using a *Power BI-specific App registration* tool provided at the following URL: <https://dev.powerbi.com/Apps>.

Similar to Dataverse, the registered external application must be granted access privileges to the cloud service – in this case, to Power BI – in order to enable the respective communication.

The only difference in the authentication code compared to Dataverse authentication would be the resource URL of the Power BI service: <https://analysis.windows.net/powerbi/api>.

Another important security option specifically for Dataverse-based solutions is authentication governance, which we look at in the next section.

## Authentication governance for internal users

Power Platform provides certain additional authentication governance features, all of which will be described in more detail in the following sections.

### Dataverse user accounts provisioning governance

In every AAD tenant, multiple Power Platform environments can be created. In fact, for large organizations that are implementing Power Platform solutions within their various business and organizational units, the number of environments can grow significantly to a potentially very high number.

Without using any specific setting, creating user accounts and assigning them Power Platform or Dynamics 365 licenses triggers the process of provisioning user account records in the `systemuser` table into all existing Dataverse environments in the tenant. To avoid this unwanted behavior, it is possible to use security groups or Office 365 Groups to manage the provisioning process and select which users will be provisioned into which Dataverse environments. To enable this feature, follow these steps:

1. Create a security or Office 365 group for every planned new Power Platform environment with Dataverse.
2. While provisioning a new Power Platform environment with Dataverse, assign the group to the environment.

3. Assign all users planned for the Power Platform environment as members in the Office 365 group.
4. Only members of the assigned Office 365 group will be physically provisioned as user records in the systemuser table in that Dataverse environment.

When a user is removed from the security or Office 365 group, their user account is automatically deactivated in the respective Dataverse environment so that they are no longer able to log in. It is worth mentioning that a user record, once created in systemuser, can never be physically deleted, only deactivated.

#### Important note



Every AAD user with the Global Administrator, Power Platform Administrator, or Dynamics 365 Administrator role will be provisioned into every Power Platform environment with Dataverse, regardless of the described setting.

## Dataverse session governance

In order to be able to enforce various AAD policies, it is important to be able to limit the session duration within Dataverse applications. By default, the Power Platform session policy depends on the general settings within AAD, which can lead to very long session durations that can go up to 90 days. In order to change this for Dataverse environments, we can define specific session timeouts by overriding the general AAD settings. The following session timeout types can be configured individually in every Dataverse environment:

- **Session timeout:** When enabled, the session timeout can be specified with values between 60 and 1,440 minutes (1 hour to 24 hours). There is also a possibility to specify a session timeout warning with values between 20 and 1,440 minutes. After the session times out, a new authentication request is enforced.
- **Inactivity timeout:** When enabled, the inactivity timeout can be specified with values between 5 and 1,440 minutes. There is also a possibility to specify an inactivity timeout warning with values between 1 and 1,440 minutes. When a user is inactive longer than the configured timeout, a new authentication request is enforced.

These settings are useful to ensure that changed authentication policies or any other permission-related changes are applied to every user within a defined timeframe.

So far in this section, we have covered the authentication of internal organizational users. Now, let's explore authentication capabilities for guest and external users.

## Azure Active Directory guest users

Microsoft Azure Active Directory allows us to invite guest users from other organizations to register within our AAD and collaborate with the organization in specified areas. This capability makes it possible for us to give external users permissions to use certain Power Platform components such as Power Apps or Power Automate flows. In order to use this capability, our own AAD must allow us to invite guests. Every invited guest must first accept the invitation they're sent in order to be physically registered in the AAD tenant. After the guest becomes part of our AAD, they can get access to Power Apps and Power Automate flows using the sharing capability. Sharing with guest users is possible, however, only in the “**user**” but not “**co-owner**” mode, so that guest users cannot modify the assigned apps or flows.

## Authenticating external users

All Power Platform components use AAD as the ultimate authentication provider, with one exception. This exception is the **Power Pages** component. This component provides a public, customer-facing portal capability that enables access to certain Power Platform content for the wider external audience.

Power Pages allows personalized access to customer-owned data such as customer profiles, customer service requests, customer field service work orders in Dataverse, and Power BI and SharePoint content. To achieve this, it is necessary to provide an authentication mechanism for external users, such as customers, partners, citizens, and others. Portal users are managed in the **contact** table in Dataverse, in which every authenticated portal user needs to have a record in this table. When it comes to authentication of these users on the portal, the following two options are available:

- **Local authentication:** The user records with the authentication artifacts (username and password) of the external users are stored in the Dataverse contact table. The authentication happens locally against the locally stored credentials. This is the standard authentication type and is available out of the box for any newly provisioned Power Pages portal.
- **External authentication:** The external user records are still stored and managed in the Dataverse contact table, but the authentication is offloaded to an external authentication provider and the authentication artifacts are not stored in Dataverse. This authentication type is disabled by default and needs additional configuration to be enabled.

In the Power Pages configuration area, we can select and configure one or more of the following authentication options:

- **Local sign-in:** Local authentication with the username/password stored in the contact table

- **Azure Active Directory:** Enables authenticating internal organization users to the portal using their AAD credentials
- **Azure Active Directory B2C:** Enables a federation solution with external authentication providers
- **Facebook:** Enables direct integration with Facebook to perform authentication using Facebook credentials
- **LinkedIn:** Enables direct integration with LinkedIn to perform authentication using LinkedIn credentials
- **Google:** Enables direct integration with Google to perform authentication using Google credentials
- **Twitter:** Enables direct integration with Twitter to perform authentication using Twitter credentials
- **Microsoft:** Enables direct integration with Microsoft to perform authentication using Microsoft credentials

It is possible to configure multiple authentication providers for a single portal, in which case external users can select their favorite provider for registration and logging in. The following screenshot illustrates this capability:

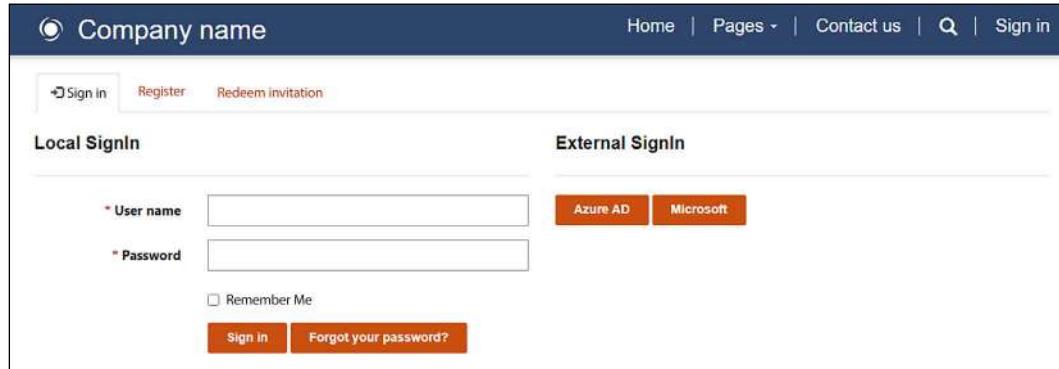


Figure 7.6: Example portal authentication configuration

As you can see, the Power Pages portal was configured to enable local authentication but also AAD authentication for internal organization users, as well as Microsoft authentication for the personal Microsoft accounts (previously known as Hotmail, Live, and Outlook) of external users.

The portal can be configured for:

- **Open registrations**, which makes it possible for everybody to register, even persons, not yet registered in the contact table in Dataverse.
- **Invitation codes**, which makes it possible to invite existing Dataverse contacts to become portal users. This method avoids duplicates in the contact table, since with every open registration, a new record is created in the contact table automatically.

An additional security option for Power Pages portals is the ability to set up IP-based restrictions to restrict the availability of the portal to certain regions.

In this section, you learned a lot about various authentication processes and the options that are available for internal and external users, as well as for service accounts. In the next section, we are going to analyze the authorization options in the different Power Platform components.

## Understanding authorization

After an interactive user or a service account has been successfully authenticated, they need to obtain certain privileges within the respective service. The Power Platform solution components provide certain individual authorization frameworks. In this section, you will learn about the authorization concepts for Dataverse, Power Apps, Power Automate, Power BI, and Power Pages.

## Authorization in Power Platform

Authorization within a Power Platform environment except Power BI is managed using environment security roles. For the purpose of understanding security roles, it is necessary to distinguish between three basic types of environments, as illustrated in the following diagram:

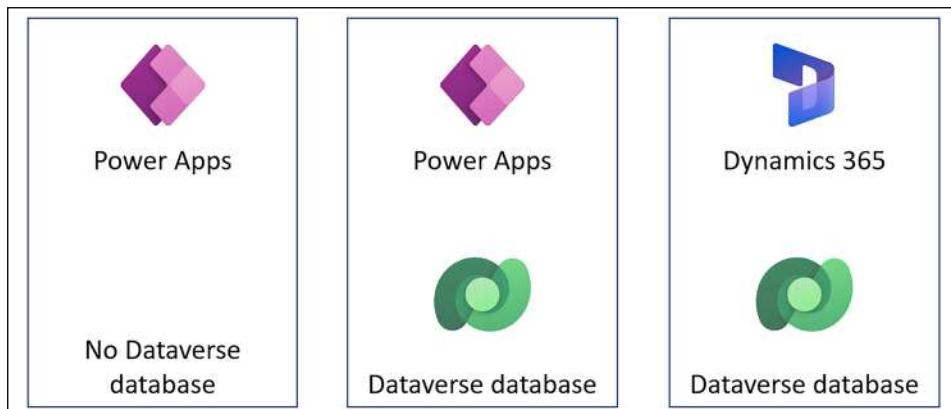


Figure 7.7: Authorization – relevant environment types

As we can see, there can be a Power Platform environment without a Dataverse database, with a Dataverse database but without any Dynamics 365 app, and an environment with some Dynamics 365 apps installed on it. Environments without a Dataverse contain only built-in roles, which cannot be modified in terms of permissions. When we create a Dataverse database within an environment, the built-in environment roles are replaced with Dataverse security roles. These security roles can be created or modified, and the permissions of the roles can be fully configured, except for the system administrator role.

The following standard environment/security roles are available in the three environment types:

- **Type 1: Environment without Dataverse:**
  - **Environment Administrator:** This role grants full privileges in the environment.
  - **Environment Maker:** This role grants creating and sharing privileges in the environment.
- **Type 2: Environment with Dataverse:**
  - **System Administrator:** This role grants full permission in the environment, such as the Environment Administrator role in environments without Dataverse. If there were previous members in the Environment Administrator role, they will be moved to this role when a Dataverse is created. The privileges in this role cannot be changed and the role cannot be deleted.
  - **System Customizer:** This role grants permissions to customize the Dataverse artifacts.
  - **Environment Maker:** This role grants application-creating privileges in the environment, similar to the same role in environments without Dataverse.
  - **Basic User:** This is the basic user role with standard access privileges to business data.
  - **Delegate:** This role grants specific permission to the user to act on behalf of another user.
- **Type 3: Environment with Dynamics 365:**
  - The workload-specific security roles are installed automatically with the respective Dynamics 365 app. The roles cover typical usage scenarios in the respective apps, for example, Sales Representative, Marketing Manager, Customer Service Representative, and Field Service Dispatcher.

After this introduction to authorization, we move forward with the authorization possibilities available in Microsoft Dataverse and model-driven apps.

## Authorization in Dataverse and model-driven apps

The authorization concept of Dataverse-based applications is very complex and consists of several authorization possibilities, of which only a part are mandatory. The additional optional possibilities can be used to implement even the most complex security requirements. In this section, you will learn the details of all the components of Dataverse authorization.

### Fundamentals of Dataverse authorization

In order to understand the Dataverse authorization concept, certain basics need to be explained first: **users and teams**, **business units**, record **ownership**, and **security roles with permissions**.

In the subsequent sections, we will take you through the details of these main components of Dataverse authorization.

### Users and teams

Users and teams are the main security principals in a Dataverse solution, and the data about users and teams is stored in the corresponding Dataverse tables. A user represents an individual interactive or technical user account. A team represents a group of users, where the team membership can be managed either manually or automatically. The difference between users and teams is that users can authenticate and connect to a Dataverse solution, while teams do not have this capability.

The following types of users are available in Dataverse:

- **Read-Write:** A read-write user type is a standard type for normal interactive users using a Dataverse application through the user interface.
- **Administrative:** An administrative user type is used for interactive users that are intended to perform administrative tasks only. The users with this user type do not have any access to business data.
- **Non-Interactive:** A non-interactive user type is used for technical user accounts for interface purposes.
- **Application:** An application user type is also used for technical user accounts for interface purposes, but the way they are created is different compared to the non-interactive type. In the first step, they are created directly in AAD, without us assigning them a license. Then, in the second step, they are created in Dataverse by providing an AAD Application ID.

- **Stub:** A stub user type does not have the ability to log in and access Dataverse. This user type is only used for importing legacy data into Dataverse when there is a need to preserve the original record ownership of former users. The stub users can only be imported into Dataverse using an *Excel-based import* process.

The following types of teams are available in Dataverse:

- **Business unit-based owner teams:** This type of team is automatically created for every business unit that's created in the Dataverse. The membership in these teams is managed automatically by the Dataverse platform. The team can have security roles assigned and can own business records.
- **Custom owner teams:** This type of team can be created manually, and the membership must be also managed manually. The team can have security roles assigned and can own business records.
- **Access teams:** This type of team can be created manually, but the real value is to use the capability of access team templates and access teams that are created automatically. The team does not have security roles and cannot own business records, but it can be used for simplified, ad hoc record-based access assignment, as described in more detail later in this section.
- **AAD Security Group teams and AAD Office Group teams:** These team types can be used for automated privilege assignment from the level of AAD, as described in more detail later in this section.

## Business units

Every Dataverse database contains a single business unit record immediately after creation. This is called the root business unit and represents the whole organization. The business unit table can be used to reflect the organizational structure of business units, departments, and teams for the purpose of hierarchical security modeling. In such a case, the hierarchical structure needs to be created, and every user needs to be assigned to a particular business unit in the hierarchy according to their position in the organization.

As an example, let us imagine an organization, interested in structuring the hierarchy according to the following diagram:

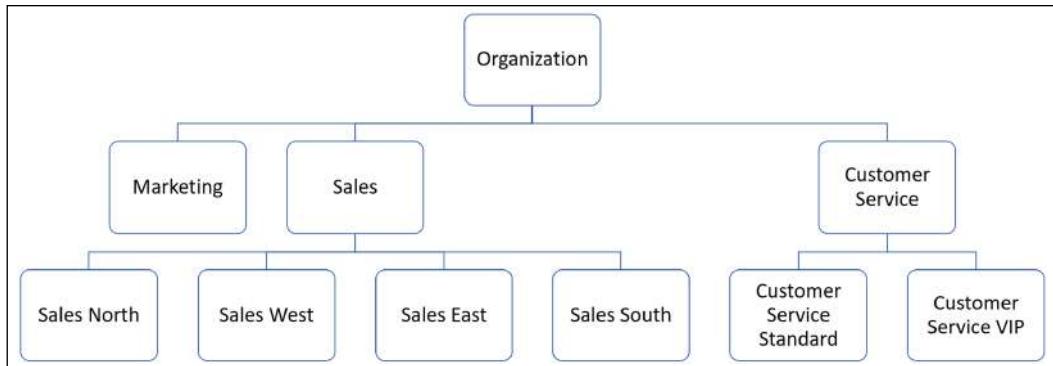


Figure 7.8 – Simplified organization structure

To map the desired structure according to the above diagram, it would be necessary to create the following records in the business units table:

The screenshot shows a user interface for managing business units. At the top, there's a header bar with "Active Business Units" and a search bar. Below the header is a toolbar with icons for New, Print, Run Workflow, Start Dialog, and More Actions. The main area is a table with columns: Name, Main Phone, Website, and Parent Business. The table contains the following data:

	Name	Main Phone	Website	Parent Business
	org819b475a			
	Customer Service Standard			Customer Service
	Customer Service VIP			Customer Service
	Sales			org819b475a
	Marketing			org819b475a
	Customer Service			org819b475a
	Sales North			Sales
	Sales West			Sales
	Sales East			Sales
	Sales South			Sales

At the bottom, there's a footer with "1 - 10 of 10 (0 selected)" and navigation buttons for Page 1.

Figure 7.9: Content in the business units table

As we can see, the business units always have a parent business unit, which represents the position in the hierarchy.

### Important note



The root business unit is always autogenerated with the same name, as the first part of the URL of the specified Power Platform environment. It is possible to change the name; however, for the root business, you cannot enter anything into the mandatory column **Parent business unit** and so you are unable to save the change. The solution for this issue is to temporarily change this column to not-mandatory, update the name of the root business unit, and then change the column back to mandatory.

## Record ownership

Every table in the Dataverse database, apart from some internal system tables, can have one of the following types of ownership:

- **Organization ownership:** Records in these tables do not have an owner; they belong to the whole organization and access management is simple: there is either access to all or no records in the table.
- **User or team ownership:** Records in these tables do have an owner. The owner can be an individual user or a team. Access management is more complex and depends on the position of the owner and the user accessing it in the business unit hierarchy.

## Security roles and permissions

Security roles define, at a very detailed level, the permissions to records in all tables, as well as table-independent permissions and privileges. For the user or team-owned tables, there is the possibility to define five different permission levels for eight different transaction types with records in the table. The permission levels, as well as the transaction types, are described in more detail in this section.

The five permission levels are illustrated in the lower part of the following screenshot of a security role called “Basic User”:

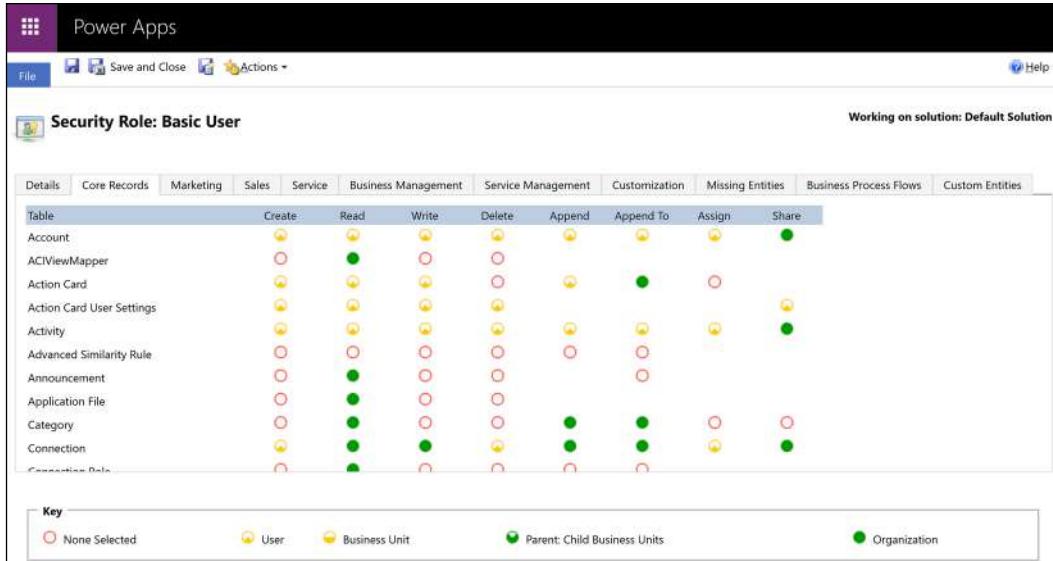


Figure 7.10: Dataverse security role-based permission levels

As shown in the screenshot, there are the following permission levels:

- **None Selected:** This permission level does not grant access to any records in the respective table or to the respective functional capability.
- **User:** This permission level grants access only to the user's own business records.
- **Business Unit:** This permission level grants access to records owned by the user, as well as to records owned by all other users assigned to the same business unit as the user.
- **Parent: Child Business Units:** This permission grants access to records owned by the user, as well as to records owned by all other users assigned to the same business unit and all *underlying* business units.
- **Organization:** This permission level grants access to all records in the respective table, regardless of the owner or the respective functional capability.

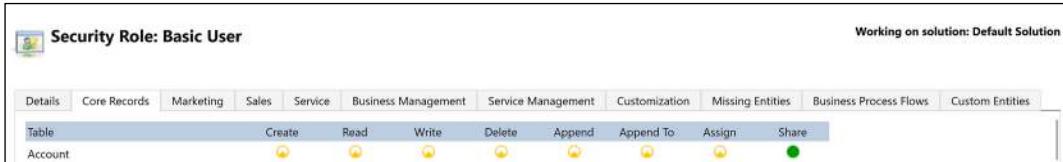
For table-based permissions, there are two types of permission levels, depending on the table ownership type:

- **Organization-owned:** For organization-owned tables, there can be only a permission type of **None Selected** or **Organization**. **None Selected** represents no access to records in the table, while **Organization** represents access to all the records in the table.

- **User- and Team-owned:** For user- or team-owned tables, all five permission levels can be specified. The impact of the selected level regarding access to records is explained in the next section of this chapter.

For table-based permissions, it is possible to specify the permissions on the level of the different transaction types, which can be performed with the records of the table.

The types of transactions that the permission can be specified for are illustrated in the following screenshot showing the upper part of the security role “Basic User”:



*Figure 7.11: Dataverse security roles – table transaction types*

As we can see, the following transaction types can be permission-specified:

- **Create:** Permission to create records in the table.
- **Read:** Permission to read records in the table. When this permission is set to **None Selected**, the user does not see any records of this table in the Dataverse application.
- **Write:** Permission to modify existing records in the table.
- **Delete:** Permission to delete records in the table.
- **Append:** Permission to attach records from this table to records in other tables.
- **Append To:** Permission to attach records from other tables to records in this table.
- **Assign:** Permission to assign ownership of records in this table to other users or teams.
- **Share:** Permission to share records from this table with other users or teams.

The configuration of table-independent permissions is much simpler and usually defines only a yes/no permission to perform certain actions in the Dataverse application.

## Setting up basic authorization

After we have learned about the basic pillars of Dataverse authorization, what needs to be done for a new user to get access to Dataverse according to their role in an organization? The following steps are to be performed in the given order:

1. Create a user account in the Azure Active Directory.
2. Assign the user a proper Power Platform license, containing access to Microsoft Dataverse.

3. Add the user to the respective environment.
4. Assign the users to the proper business unit in the existing hierarchy.
5. Assign the user one or multiple security roles.

Now it is time to learn the key information – how the standard hierarchical Dataverse security actually works.

## Standard role-based security

Standard role-based security is mandatory and must be configured for every Dataverse application to allow access to data and business capabilities for the users. This security model makes use of the **Users** owning **Business Records**, which are assigned to **Business Units** and have one or more **Security Roles** assigned to them, as shown in the following simplified diagram:

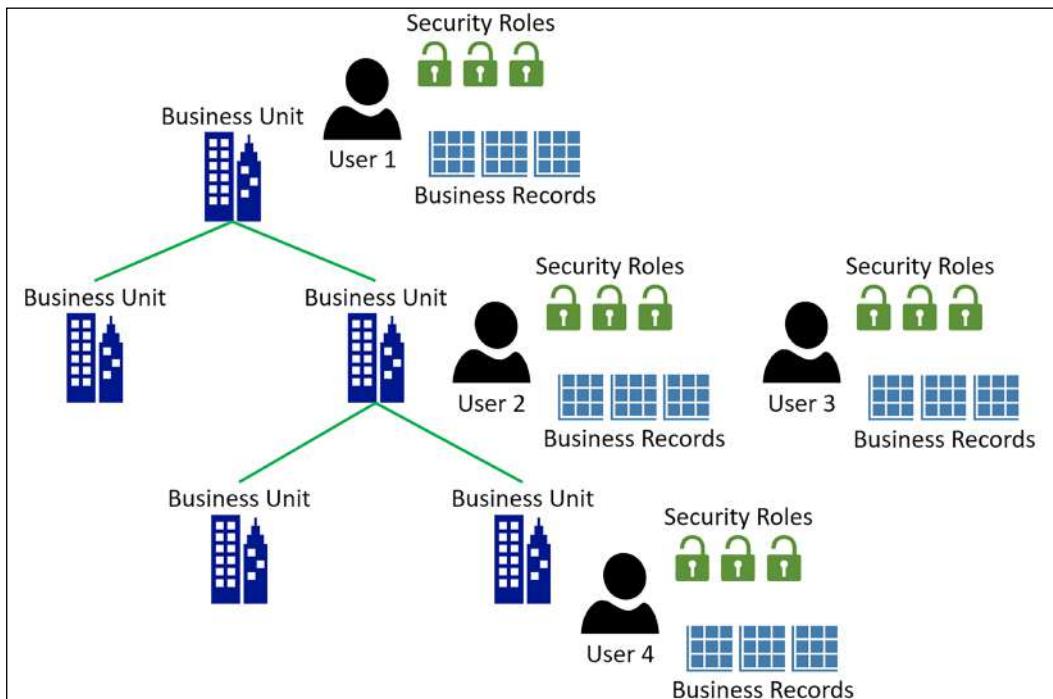


Figure 7.12: Standard role-based security

As we can see, there is a hierarchical structure of business units, where every user is assigned to one of the business units and every user has one or more security roles assigned to them. Every user is an owner of certain business records in various tables. The authorization to see and manage data and to perform certain actions in the Dataverse application depends on the following:

- The level in the business unit's hierarchy that a user is assigned to
- The assigned security roles to the user
- The ownership of records by the user and other users in the hierarchy

Let's explain this capability as an example for **User 2** while considering only the **Read** transaction type for a certain business table:

- **Option 1:** The user has the **None Selected** permission level in their security role for the table. In such a case, the user will have no access to any records in the table.
- **Option 2:** The user has the **User** permission level in their security role for the table. In such a case, the user will only see their own records.
- **Option 3:** The user has the **Business Unit** permission level in their security role for the table. In such a case, the user will see their own records, as well as the records owned by **User 3**.
- **Option 4:** The user has the **Parent: Child Business Units** permission level in their security role for the table. In such a case, the user will see their own records, as well as records owned by **User 3** and **User 4**.
- **Option 5:** The user has the **Organization** permission level in their security role for the table. In such a case, the user will see their own records, as well as records owned by all other users in the organization, regardless of their position in the business unit hierarchy.



#### Important note

A user can be moved from one business unit to another. In such a case, the user will lose all assigned security roles, and the assignment of the proper security roles must be performed again in the new business unit.

## Modernized Business Units

An alternative method of handling the standard security model was introduced recently under the name "Modernized Business Units." In order to understand this method, it is necessary to mention one important fact about security roles and business units.

### Important note



Whenever there is a new security role created in Dataverse, it is created within a selected business unit. After being created, the security role is replicated downstream in the hierarchy, so that in every underlying business unit, there is a copy of the security role. This copy is read-only and represents local permissions on the level of the current business unit. Typically, security roles are created on the root business unit level, but there is an option to create them on any level in the hierarchy.

With modernized business units, the barrier between business units is broken and it is possible to assign security roles from **different business units** to a user, and so granting access to data from various parts of the organization hierarchy.

This capability can also improve the **Group Teams** capability, described later in this section. Since the Group Teams capability does not assign users to business units, we can overcome this limitation by assigning security roles from different business units to the group teams.

In the following sections, you will learn about the additional, optional authorization options within a Dataverse-based solution.

## Group teams

AAD, together with Dataverse, provides an alternative, simplified method of granting access rights within Dataverse applications: using group teams. The process of creating a group team consists of the following steps:

1. Create a security group or Office 365 group in AAD and make note of the **ObjectID** of the group.
2. Create a Dataverse team of the **AAD Security Group** or **AAD Office Group** type – depending on the AAD group type, and enter the **ObjectID** you specified previously to create a relation with the AAD group.
3. Assign security roles to the Dataverse team.

After such a team is created, new users can be empowered to have appropriate access to data in the Dataverse application by adding those users as members to the AAD group in the AAD, without the need to manually assign them security roles inside Dataverse.



#### Important note

This alternative approach does not assign a user automatically to a business unit, so it works fully automatically only if every user remains in the root business unit. If an established business unit hierarchy is used, every user would first need to be moved to a business unit, and only then can the described capability be applied.

A different approach can be to use the Modernized Business Units capability as described above.

## Authorizing model-driven app access

The first level of authorization, before a user can even start working with a model-driven application, is authorization at the app level. Dataverse and model-driven apps have a one-to-many relationship, so within a single Power Platform environment containing a Dataverse database, many individual model-driven apps can be created. In order to manage access to these apps, there are two methods that can be used:

- We can **assign security roles** individually to each model-driven app, in which case, only users holding some of those roles would get access.
- We can **share** model-driven apps with users, security groups, or Office 365 groups and so grant access to the users or members of the groups.



#### Important note

The second described method using sharing can be used to grant access to model-driven applications to Azure Active Directory guest users.

Assigning security roles is illustrated in the following screenshot:

The screenshot shows the Microsoft Power Platform Admin Center interface. On the left, there is a grid of four model-driven apps: 'Dynamics 365 App for Outlook' (blue D icon), 'Field Service' (yellow hexagon icon), 'Portal Management' (blue square icon), and 'Project' (green document icon). Each app has a brief description and the 'UNIFIED INTERFACE' label at the bottom. To the right of the grid is a modal window titled 'Manage Roles - Field Service'. The window contains instructions: 'Choose an app URL that is easy to remember and then select which roles can access it.' It has two sections: 'App URL Suffix' (which is collapsed) and 'Roles' (which is expanded). The 'Roles' section displays a table with two columns: 'Name' and 'Business Unit'. All roles listed have the value 'robertrybaric' in the 'Business Unit' column. The roles listed are: Account Manager, Activity Feeds, AIB Roles, AIB SML Roles, Analytics Report Author, App Deployment Orchestration..., App Profile Manager Administrator, App Profile User, Approvals Administrator, Approvals User, Basic User, BizQAApp, Bot Author, Bot Contributor, and Bot Transcript Viewer.

Name	Business Unit
Account Manager	robertrybaric
Activity Feeds	robertrybaric
AIB Roles	robertrybaric
AIB SML Roles	robertrybaric
Analytics Report Author	robertrybaric
App Deployment Orchestration...	robertrybaric
App Profile Manager Administrat...	robertrybaric
App Profile User	robertrybaric
Approvals Administrator	robertrybaric
Approvals User	robertrybaric
Basic User	robertrybaric
BizQAApp	robertrybaric
Bot Author	robertrybaric
Bot Contributor	robertrybaric
Bot Transcript Viewer	robertrybaric

Figure 7.13: Assigning security roles to model-driven apps

As we can see, we can select all the necessary security roles and assign them to the respective model-driven app. This approach makes it possible to limit the number of model-driven apps that are visible and available to users so that they only see those relevant to them.

**Important note**

A Dataverse system administrator always has access to all existing model-driven apps in a Power Platform environment.

## Hierarchy security

**Hierarchy security** is an additional and optional security model that allows organizations to configure a second dimension of hierarchical security configuration. Hierarchy security can be implemented using the following approaches:

- **Manager hierarchy:** This security model uses the manager hierarchy in the organization, where every Dataverse user can have another Dataverse user assigned as their manager.

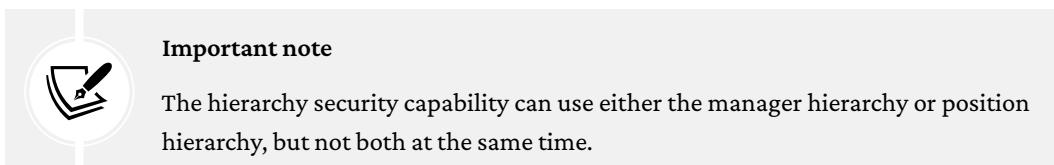
The manager usually has read-write access to the records of their direct reports and read-only access to records owned by users below their direct reports level. The model also makes it possible to specify the level's depth, which is where the manager has access to records in the hierarchy.

- **Position hierarchy:** This security model uses an additional structure of positions, which can be defined independently of the business unit hierarchy, as well as of the manager hierarchy. After the position's hierarchy has been created, individual users can be assigned to the respective positions.

Users in higher positions have read-write access to records owned by users directly below and read-only access to records owned by users lower in the position's hierarchy.

This approach can be very useful for matrix organization, or any other situation where the standard line management structure is not enough to model the real security needs. A good example is the professional services industry, where the majority of employees are working in project mode. In such a case, a position hierarchy could be useful to consider in the following way:

- Line management hierarchy implemented with standard role-based security
- Project organization hierarchy implemented with the position hierarchy



In the following sections, you will learn about two options you can use to grant ad hoc access to records.

## Record sharing

**Record sharing** is, together with standard role-based security, the oldest security model in Dataverse since the first versions of Dynamics CRM. Record sharing makes it possible to manually or automatically share a record with other users or teams. This sharing creates a full principal object access structure in the Dataverse database, so it is considered performance heavy. Heavy use of record sharing is discouraged and, generally, record sharing should be replaced with the *access teams capability*.

## Access teams

The **access teams**-based security model is a good modern alternative to record sharing. This approach makes it possible to provide temporary, ad hoc, record-based access rights to individual users or teams. To set up access teams for a certain table, follow these steps:

1. Enable the access teams capability for a business table in the Power Apps maker portal.
2. Create an access teams template for the table, defining the required privileges (read, write, delete, assign, share, append, and append to).
3. Place an access teams subgrid on the table form.

After completing these steps, every user that has full access to the record can manually add another user to the access team's subgrid and so grant the specified privileges for that record to the user. The Dataverse platform auto-creates an access team for the record after the first user is added to enable the sharing capability. Deleting a user from the subgrid revokes the granted privilege from the user.

The benefits of this security model, specifically compared to record sharing, are as follows:

- Access teams-based access is **lightweight** since it does not create full principal object access structures in the Dataverse database and so has a minimal performance impact on the Dataverse.
- Access teams are better to manage since the granted access privileges are visible on the user interface and can be easily revoked manually as well as automatically, for example, based on record status change.

All the previously described security models are considered record-level security since they are offering various options to manage access to individual records in the Dataverse database. In the next section, you will learn about the possibility to protect data at the individual column level.

## Column-level security

There are often requirements to protect individual columns in the Dataverse records, specifically, highly sensitive columns such as usernames, passwords, and banking account or credit card details. Protecting individual columns can be implemented using **column-level security**. To set up this capability for a certain column, follow these steps:

1. Enable **column-level security** for a column in a table in the Power Apps maker portal.
2. Create column security profiles to specify column permissions (read, create, and modify) for the protected columns. This can be done in the Power Platform Admin Center.
3. Add users or teams as members to the column security profiles to grant the appropriate access. This can also be done in the Power Platform Admin Center.

After completing these steps, users not assigned to any column security profile will not have access to the protected columns in any part of the specific application. This also applies to the specific API, which also protects the content of those columns based on the user credentials used in the API.

## User interface security

The roles-based security model of using security roles also applies to certain user interface components in model-driven apps, specifically:

- **Model-driven forms:** For table forms, there is a possibility to switch role-based security on in order to enable/disable access to the table forms for various user groups based on security roles. This capability is available only for the *main* table forms. It is important to always specify one table form as a *fallback* so that every user always has at least one table form available to work with. This capability makes it possible to provide the best user experience to various user groups. We can do this by providing them with tailored table forms that contain the data that's relevant to their job roles.
- **Model-driven dashboards:** For all types of dashboards (standard, interactive, and table-specific dashboards), there is a possibility to switch role-based security on in order to enable/disable certain dashboards for various user groups. For dashboards, there is also the capability to define *fallback* dashboards, which is particularly important for table-specific dashboards so that a dashboard is always visualized on the table form.

In the previous sections, we covered all the authorization possibilities of Dataverse-based applications. Next, you will learn about authorization within canvas apps.

## Authorization in canvas apps

Canvas apps are very different from model-driven apps in terms of authorization. For canvas apps, the following authorization concepts apply:

- Authorization of apps
- Authorization of connections

In the following sections, we will cover these authorization concepts in more detail.

### Authorization of apps

Every canvas app is created by a particular user with the proper permissions, where that user is the owner of the app. In order to allow other users to use the app, the owner can share the app with individual users or groups of users. There are the following possibilities for sharing:

- **Share with everyone:** This setting makes the app automatically available to every user in the AAD tenant who has an appropriate Power Platform license.
- **Share with a security group:** This setting makes the app available to every member of the selected AAD security group. All users joining the security group will automatically inherit the permissions for the app.
- **Share with individual user:** This setting makes the app available to individual users in the AAD tenant.

There are the following types of sharing:

- **User sharing:** This type of sharing grants the user permission to use the canvas app.
- **Co-owner sharing:** This type of privilege grants the user permission to use and also modify the app.

Canvas apps can be shared not just with full AAD users but also with **guest users**. Guest users, however, can only be granted the user permission level, not the co-owner permission level.

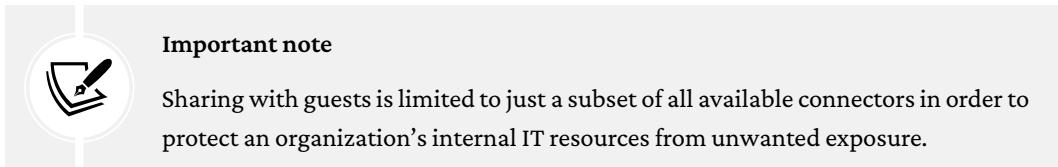
Part of this sharing capability is also to specify whether the platform should send the new users of the app an invitation email to notify them that a new app is ready for them to use.

### Authorization of connections

The main benefit of canvas apps is the possibility to connect to a wide variety of Microsoft, as well as third-party, systems using public or custom connectors. For the canvas app to work properly, the user of the app must be granted the appropriate privileges to all underlying systems that the app connects to through the use of connectors.

So, the second step in granting authorization for canvas apps is to authorize the connections. Since there is a large number of public connectors and the number is still growing, there is no unified way to configure authorization; rather, it fully depends on the underlying technology.

For the Dataverse connector, for example, there is the possibility to assign appropriate Dataverse security roles to the users during the sharing process. Other connector types are implicitly shared, but for some connectors, specifically third-party connectors, authorization needs to happen within the respective third-party system individually. The same applies to connectors to on-premises systems using the on-premises data gateway.



#### Important note

Sharing with guests is limited to just a subset of all available connectors in order to protect an organization's internal IT resources from unwanted exposure.

## Authorization in Power Automate

When thinking about authorization within Power Automate, it is important to distinguish between the two types of Power Automate flows: **background** flows and **interactive** flows.

In the following sections, you will learn about the authorization concepts for the two main Power Automate flow types.

### Authorization of background flows

Background Power Automate flows (automated flow and scheduled flows) are triggered automatically and usually do not interact with the end users. Since Power Automate flows use the same concept of connectors that canvas apps do, the only authorization that has to be configured is the authorization for the connections.

The flow author must provide proper authorization for all connections used in the flow so that the flow has permission to perform the required transactions in the connected systems. Background flows can, however, be shared with other users as co-owners, to give them the ability to manage or modify the flows in the same way as the owner.

## Authorization of interactive flows

The only interactive Power Automate cloud flow type is the button flow. Making a button flow available to other users in the organization is done in a similar way as it is for canvas apps – using sharing. A button flow can be shared with other users, and the owner can specify whether the connections used in the button flow will be authorized with the credentials of the flow owner or with the credentials of the flow user. It is also possible to specify co-owners for button flows so that they can manage or modify the flows.

## Authorization of desktop flows

A specific category of interactive flows is **Power Automate Desktop flows**. Since the desktop flows do not use any connectors or any kind of API communication, any authorization of the automated underlying IT systems must be wired into the flow as sign-in information that is presented to the automated systems.

## Authorization in Power BI

As we've already mentioned several times in the previous chapters, Power BI is technologically very different from the other Power Platform components, and so is the authorization concept of Power BI. In this section, we will focus on the authorization topics relevant to Power Platform.

To understand the authorization, it is important to understand the dataset and query types used in Power BI first. There are two major dataset and query types:

- **Import:** This type of dataset and query generally stores the metadata and actual data of the dataset physically on the Power BI platform – a copy of the data is created. The dataset is equipped with the credentials of the dataset creator. When the creator shares the dataset along with the reports, dashboards, and other Power BI elements, the credentials of the creator are used by all the other users.
- **DirectQuery:** This type of dataset and query stores only the metadata physically on the Power BI platform, but the data is retrieved from the data source on the fly upon opening the Power BI visuals (reports and dashboards). Based on the configuration, it is possible to automatically forward the user credentials of users running the Power BI solution to the underlying data source. In this case, the authorization for accessing the data is provided by the underlying system.

Another main Power BI concept to understand is **row-level security (RLS)**. The main purpose of RLS is to define roles with security restrictions, specified using the DAX filter expressions. The following screenshot illustrates a simple RLS role created for a Dataverse dataset:



Figure 7.14: Power BI role-level security example

As you can see, the role "Seattle" filters all accounts and contacts in the Dataverse to those where `city = "Seattle"`.

## Authorization in Power Pages

Power Pages is a Power Platform component that exposes certain portal capabilities and Dataverse data to external audiences, such as customers, partners, or citizens. It is necessary to be able to configure the following:

- Which content (navigational areas or web pages) will be available to anonymous public users?
- Which content (navigational areas = web pages) will be available to authenticated users?
- Which Dataverse data types (tables) and individual records will be available to authenticated users?

These content- and data-based privileges are managed by a structure of permission-related settings, as illustrated in the following diagram:

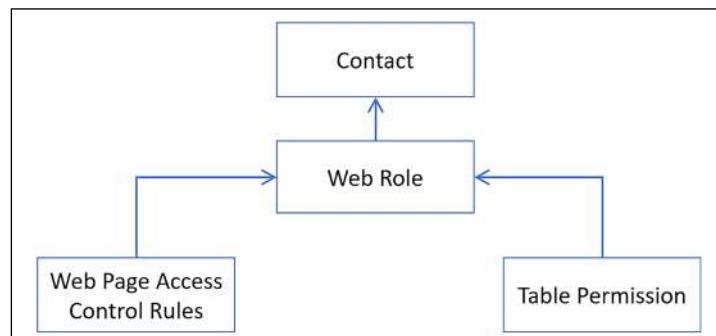


Figure 7.15: Power Pages portal authentication structure

As we can see, the security principal for Power Pages portals is always the **Contact** table. The main authorization table in the structure is the **Web Role**; however, the actual permissions are specified in the other two tables. The following are the roles of the tables according to the previous diagram:

- **Contact:** The contact record represents the actual user of the portal. Every contact can be assigned one or more web roles.
- **Web Role:** The web role is the main container for all privileges-based settings. Every web role can be assigned one or more web page access controls rules and table permissions.
- **Web Page Access Control Rules:** The web page access control Rules specify permissions for individual web pages within the Power Pages portal instance. It is possible to grant a change (for administrators and designers) or read (for users) permission. Without this permission, the specified web page would not be available to the portal user.
- **Table Permission:** The table permission specifies detailed permissions on the level of the Dataverse table. This permission defines the scope of records available to the user (records related to the contact, to the parent account of the contact, and global scope) and the privileges on the table records (read, write, create, delete, append, and append to).

In this section, we have provided a detailed overview of the authorization possibilities of the Power Platform components. You have learned about the various authorization options available within Dataverse, as well as how to set up authorization for the other cloud services within the Power Platform product family.

In the next section, you will learn about Power Platform's compliance, privacy, and data protection features.

## **Understanding compliance, privacy, and data protection**

In this section, you will learn how Microsoft cloud services, including Power Platform, handle compliance, privacy, and data protection requirements.

Compliance, privacy, and data protection within the Power Platform is handled by the same standards as the other Microsoft cloud offerings, that is, Microsoft Azure and Microsoft 365. Microsoft provides certain centralized information sources, repositories, and tools to inform customers about what data is processed, where the data is stored, how the data is protected, and what global, regional, national, and industry-specific standards are supported. The following are the most important information sources and tools:

- **Microsoft Privacy Statement:** Specifies which personal data is processed, how, and for what purpose. The privacy statement can be found at the following link: <https://privacy.microsoft.com/en-us/privacystatement>.
- **Microsoft Trust Center:** Provides a rich set of information about security, privacy, and compliance within all Microsoft cloud services. The Trust Center can be found at the following link: <https://www.microsoft.com/en-us/trust-center>.
- **Microsoft Service Trust Portal:** Provides a large collection of audit reports from independent international and other authorities about compliance with data protection standards and regulatory requirements. The Service Trust portal can be found at the following link: <https://servicetrust.microsoft.com>.
- **Compliance Manager:** Provides a personalized compliance assessment and checklist of action steps necessary to achieve the required level of compliance. The Compliance Manager is a private tool, available only to existing Microsoft cloud customers, and can be found at the following link: <https://servicetrust.microsoft.com/ComplianceManager/V3>.

Specifically for Power Platform, the following data protection standards are implemented:

- Power Platform customers have the possibility to choose any Power Platform-enabled region to create their environments in. This allows them to fulfill the possible data residency requirements.
- Power Platform data, specifically data within Dataverse databases, might be replicated for failover safety purposes to other regions, but always within the same geography.
- Power Platform communication is encrypted in transit using the *TLS 1.2 or higher* network security protocol. Unencrypted HTTP communication is not possible.

- Power Platform data, specifically data in the Dataverse databases and metadata in Power BI, is encrypted at rest using **SQL Server Transparent Data Encryption (TDE)**. As standard, the encryption keys are managed by Microsoft, but there is a possibility for customers of a certain size (currently with 1,000 or more Power Apps or Dynamics 365 licenses) to leverage the **bring-your-own-key (BYOK)** capability. Using this feature, customers can purchase or generate their own encryption keys and use them to encrypt their Dataverse database. The own key is stored within Azure Key Vault for enhanced protection.



#### Important note

The term **encryption at rest** represents the encryption of data stored in permanent storage, like a database file stored on a **hard drive (HDD)** or **solid-state drive (SDD)**.

- Power BI data for **import** datasets and queries is stored in Azure Blob storage and encrypted with encryption keys managed in Azure Key Vault. Power BI Data for **DirectQuery** datasets and queries is not stored at rest within Power BI but stays in the original data source.
- Power Platform provides a set of procedures and tools used to address typical **General Data Protection Regulation (GDPR)**-related requests. There are possibilities provided on the Power Platform administration portals, through PowerShell or using APIs, to locate, modify, delete, or export personal data from the respective components of the Power Platform ecosystem.

In this section, you have learned some basic information about compliance, privacy, and data protection in Power Platform. In the next section, we are going to present proven best practices for establishing a mature security model.

## Presenting security best practices

In this section, you will make yourself familiar with some best practices for building a robust and mature Power Platform security model. Following the best practices can ensure that your Power Platform solution fulfills the necessary security standards and that you are building a solution that can be maintained and upgraded in the future.

## Dataverse security roles

In this section, we will present some proven best practices for designing, implementing, and using Dataverse security roles.

### Modifying security roles

Security configuration in Dataverse applications can be very simple, where the default Dataverse or Dynamics 365 security roles can be used without any modification. However, if the security requirements are advanced, there might be a need to tailor the permissions provided in the standard roles. In such a case, it is highly recommended **not to modify** the default security roles, but rather to create new custom roles. The best method is to select the best suitable standard security role, make a copy of it, modify the permissions as needed, and then use the custom role instead of a standard role.

### Layering of security roles

Every user in a Dataverse-based application must have at least one security role assigned, but there is no restriction on how many security roles can be assigned to an individual user.

Configuring individual security roles for every required permission combination is not always the best way, since it might require creating a multitude of similar security roles, which could make it challenging to maintain. An alternative approach to a situation where there are many required permission combinations and most of the standard security roles cannot be used is to use security roles layering, as illustrated in the following diagram:

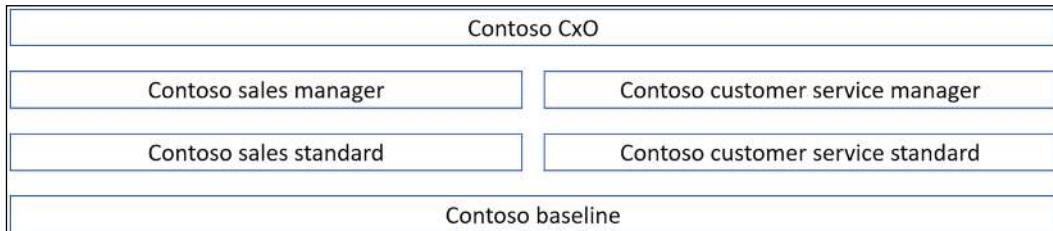


Figure 7.16: Security roles layering

As we can see, every user in the organization will be assigned a certain number of security roles with additive permissions, based on their level in the organization. The configuration in this example is as follows:

- **Contoso baseline:** This security role will specify basic permissions for every individual user in the organization.

- **Contoso sales standard:** This security role will specify additional permissions for every sales representative in the organization.
- **Contoso customer service standard:** This security role will specify additional permissions for every customer service representative in the organization.
- **Contoso sales manager:** This security role will specify additional permissions for every sales manager in the organization.
- **Contoso customer service manager:** This security role will specify additional permissions for every customer service manager in the organization.
- **Contoso CxO:** This security role will specify additional permissions for the members of the top-level management of the organization.

This approach is beneficial in different ways. These benefits are as follows:

- Every individual security role above the baseline specifies only the additional permissions against the baseline or lower-level security roles. This ensures that the security roles' configuration is easier compared to the standard security roles approach, where every role contains the full set of permissions needed for the role.
- If a general permission change is required, this change can be performed in a single security role instead of several of them.
- Generally, this approach would require fewer security roles to be created and maintained.

In case the permission requirements are only slightly different from the standard security roles, it can be beneficial to keep using the standard security roles, but to also create one or a few additional custom security roles that extend the permissions of the standard roles.

## Dataverse content-based security

All the Dataverse security models described in this chapter are user-centric, which means the privileges that are granted are always inherited from the user accessing Dataverse. There are, however, situations where content-based security could be required. A typical example could be a Dataverse solution for a bank, where all private customer records are stored in the standard Contact table. Let's assume 98% of the private customers are usual customers and the last 2% are VIP customers. While the data of the usual customers can be seen by every user within the standard role-based authorization model, for the VIP customers, certain data should be available to only a small group of users.

There are several possibilities to solve this content-based security requirement, as described in the following sections.

## Using business units

Using business units is the simplest approach as we only leverage standard Dataverse capabilities. The essence of this solution would be to create an appropriate business unit hierarchy and offload the ownership of the sensitive business records to a user or team, assigned to a business unit that can't be accessed by the vast majority of the Dataverse application users.

The benefit of this approach is that it uses standard Dataverse capabilities and that access to the protected data is reliably restricted in all parts of the Dataverse solution.

The drawback of this approach, however, is that all the data is hidden from the standard users, not just the sensitive parts.

## Using table form switching

Using table form switching is a frontend-only solution that leverages the capability of having more than one main form per table. Instead of managing access to table forms based on security roles, this approach would switch table forms based on content during the onloading event using JavaScript event handling.

The benefit of this approach is the selective show/hide capability of sensitive information based on content, by just configuring the table forms with exactly the required content.

The big disadvantage is that this approach is not 100% safe, since skilled users can figure out how to get access to the hidden pieces of information using the advanced find capability, or other similar approaches.

## Using client-side scripting or business rules

Client-side scripting or business rules can to some extent achieve the same as table form switching with all the benefits and disadvantages.

## Using server-side event handlers

Using server-side event handlers is a backend solution that leverages the capability of creating *Dataverse plug-ins*. The essence of this solution would be to implement server-side event handlers that hide/mask some sensitive data based on the content of the business record. These event handlers would need to be registered for both the **Retrieve** and **RetrieveMultiple** Dataverse events to protect the sensitive data in both table views as well as table forms. This approach requires custom development and protects the sensitive data reliably across all parts of the Dataverse solution, with the exception of the standard Dataverse reports based on Microsoft **SQL Server Reporting Services (SSRS)**.

In the following section, we will discuss some additional ways to achieve a consistent authorization solution across various cloud components used within a Power Platform solution.

## Integrate security across solution components

A Power Platform solution usually consists of several Power Platform and other components, from which some are directly integrated at the level of individual Dataverse business records. The best example of such a solution would be a Dataverse solution that's been extended with SharePoint for integrated document management and Power BI for analytics and reporting, as shown in the following diagram:

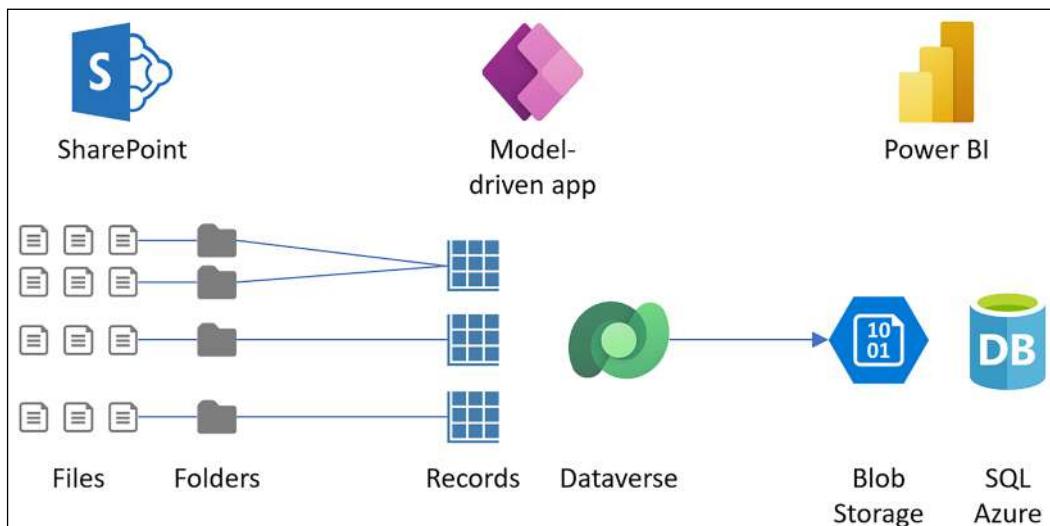


Figure 7.17: Power Platform solution example

As we can see, the standard integration approach works as follows:

- For every SharePoint-enabled Dataverse table, there can be one or more SharePoint folders where users can upload files and work with them from within the context of the Dataverse record in a model-driven app. For this standard integration to work, every Dataverse user would need to have proper permission on the whole **SharePoint** site collection, integrated with Dataverse.
- A standard Power BI solution for Dataverse performs an import of the selected Dataverse records into the Power BI internal storage (**Azure Blob Storage** and **Azure SQL Database**) and uses this local copy of the Dataverse data as a dataset for analytics and reporting. The dataset is equipped with the credentials of the creator, who has full permission to all the records in the Dataverse table.

Both of these standard solution approaches do not respect the very detailed authorization concept of the Dataverse applications and, instead, use their own not-integrated authorization concept for accessing data – folders and files in SharePoint and business data in Power BI reports and dashboards. The security consequence of not implementing any additional security solution would be as follows:

- Every Dataverse application user would be able to switch over to the integrated SharePoint site collection and find any files belonging to any Dataverse records they have no permission to see within Dataverse.
- Every Dataverse application user would see all the Dataverse data in the Power BI reports and dashboards, to which the credentials of the creator of the Power BI datasets grant permissions.

If there is a requirement to respect Dataverse-based access permissions to the folders and files in SharePoint and business data in Power BI, an additional layer of security needs to be implemented, as described in the following sections.

## Dataverse-SharePoint integrated security

Achieving a consistent and automated authorization solution for Dataverse and SharePoint is possible using some of the following options:

- **Full access:** The standard setup for Dataverse-SharePoint integration requires giving respective contributor permissions to the whole SharePoint site collection, which is used for integration with Dataverse. This is the simplest solution, but it opens up a potential security hole, as described earlier in this section.
- **Manual access control:** This approach would require not granting site collection-wide permissions to all Dataverse users, but rather granting access to individual folders and files in the SharePoint site collection hierarchy manually. This could be done by an administrator. While this is theoretically possible, for a larger number of Dataverse business records, this would be impossible to manage.
- **Permission replication development:** This approach would require building a custom solution for Dataverse to replicate the Dataverse record permissions in the integrated SharePoint folders and files. This would require significant custom development effort to cover all the very specific Dataverse authorization possibilities and their changes over time.
- **Third-party solutions:** There are commercial solutions on the third-party market that cover permission replication from Dataverse to SharePoint.

## Dataverse-Power BI integrated security

When integrating Power BI with Dataverse, there are also some ways to achieve consistency when accessing data:

- **No authorization:** This type of authorization means that Power BI will present all the Dataverse data in the visuals to all users. This is a simple and viable option when providing unfiltered data to everybody is acceptable. This authorization can be implemented using both *import* as well as *DirectQuery datasets*. The only necessary configuration setting is to ensure that the credentials of the datasets allow full access to all the data in all the tables being used in the Power BI visuals.
- **Static authorization:** This option is a modification of the previous one. We would need to specify dataset credentials that do restrict access to some of the Dataverse data, but the result would present the same statically pre-filtered subset of data to every user in the Power BI visuals.
- **Row-level security authorization:** This type of authorization would require a parallel authorization level to be built along the Dataverse authorization. This can be an intended approach when the Power BI visuals should present data that's been filtered using different criteria than what is implemented in the Dataverse authorization. This authorization can be implemented using both *import* as well as *DirectQuery datasets*. The configuration would require the use of credentials for the datasets that allow full access to all the data in all the tables being used in the Power BI visuals, as well as creating the respective RLS roles and assigning them to Power BI users.
- **Dataverse authorization:** This type of authorization can be implemented using the *DirectQuery* dataset type only. The Power BI visuals would then display the same data to the Power BI users that they can see within Dataverse applications. The only necessary configuration setting is to ensure that the credentials of every individual user will be propagated to the data query when data is being retrieved from Dataverse.

In the following sections, we will discuss the topic of automating the whole identity and access management process once more, but this time, we will include the ability to integrate into an on-premises Active Directory.

## Using identity and access management automation

Large organizations with complex IT ecosystems usually use **identity and access management (IAM)** solutions to provision new users and manage user permissions within existing IT systems. It is important to have the capability to automate these processes for complex Power Platform-based solutions as well, so as to be able to support the existing customer's IAM solution.

This topic was already touched upon in *Chapter 3, Understanding the Microsoft Power Platform Architecture*, so now we will extend this to scenarios that cover Active Directory (AD) integration. For all AD integration approaches, there is an additional component called **AAD Sync** (part of **AAD Connect**) that's deployed. This synchronizes the AD user accounts in AAD. An IAM automation solution would need to consider this feature and work differently compared to a solution for pure cloud identities, as illustrated in the following diagram:

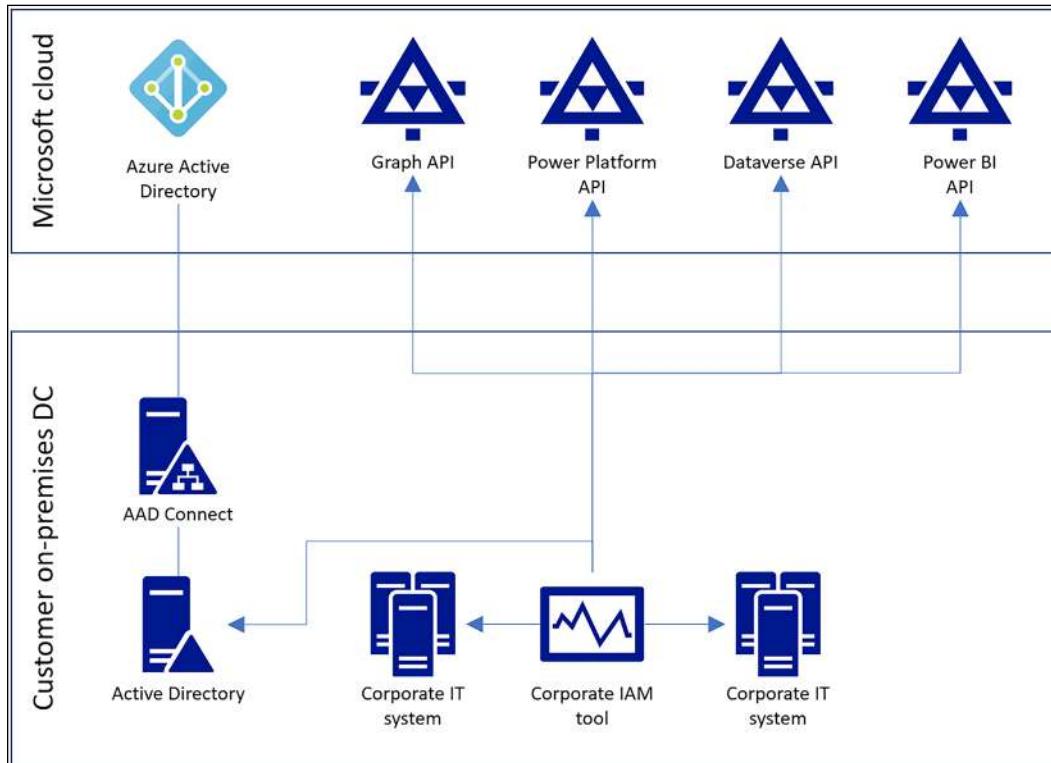


Figure 7.18: Identity and access management automation

As we can see, the first step in provisioning a new cloud user account in AAD is now fully under the control of AAD Connect, specifically the AAD Sync service. The AAD Sync service is a service that's triggered with a timer, usually every 15 minutes, which then performs all pending user account synchronizations. In order to fully automate the IAM process for Power Platform, we need to follow these steps, which need to be implemented within the customer's IAM solution:

1. Create the new user identity in the on-premises Active Directory and flag the new user account as cloud-enabled. After this, AAD Sync will recognize this new AD user for synchronization in AAD.

2. Start a polling subprocess to poll AAD using the *Graph API*, to figure out when the user account is already provisioned in the AAD.
3. OPTIONAL: After the new user account has been provisioned in AAD, assign the user to all necessary security/Office 365 groups associated with the Dataverse environments where the user should be granted access.
4. Assign the necessary Power Platform licenses to the new user using the *Graph API*.
5. Since provisioning the new user in the Dataverse environments is also an asynchronous process, start a polling subprocess to poll all the Dataverse environments using the *Dataverse API*, to wait for the automated user creation. Alternatively, create the users in Dataverse actively, using the *Dataverse API*.
6. In every Dataverse environment where the new user is already provisioned, assign the necessary authorization privileges and permissions using the *Dataverse API*.
7. Assign the necessary authorization privileges in Power BI using the *Power BI API*.

With all the information we've gathered from the previous sections, we are now ready to finally shape the Power Platform mature security model.

## Establishing the Power Platform mature security model

With the help of the security concepts of Power Platform and its components, it is possible to establish a mature security model at all possible levels, as illustrated in the following diagram:

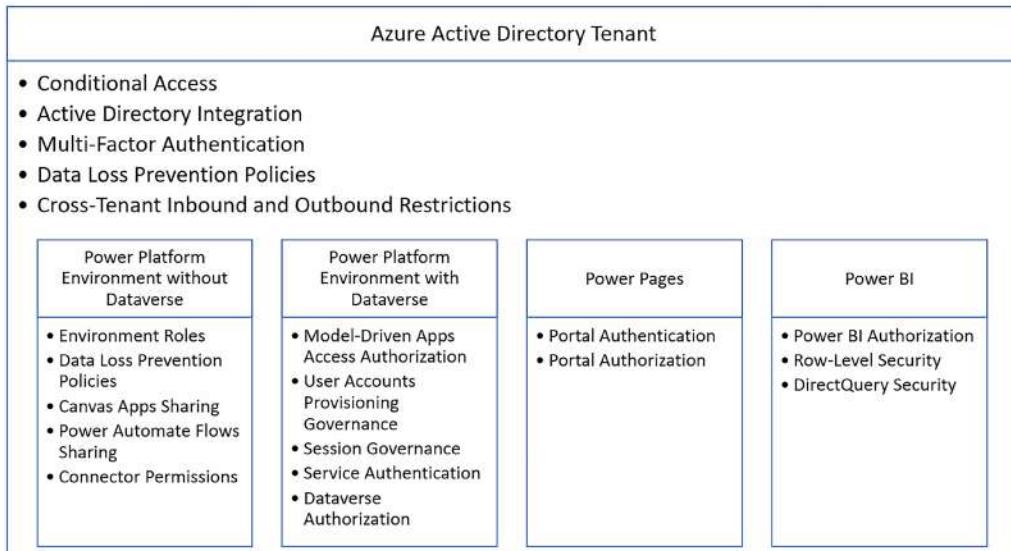


Figure 7.19: Mature Power Platform security model

As we can see, there are certain security options available on the whole **Azure Active Directory Tenant**, and others are available within the different Power Platform components. The various options can be used for the following purposes:

- **Conditional access:** Use AAD conditional access if you need to generally restrict access to the whole AAD tenant and all the components within the tenant, including Power Platform, for users based on defined criteria (geography, device, application, and membership).
- **Active Directory integration:** Use Active Directory integration to ensure centralized password control, centralized identity policies, and single sign-on.
- **MFA:** Use MFA to increase the security of the authentication process and prevent malicious authentication attacks.
- **Data loss prevention policies:** Use data loss prevention policies to prevent unattended business data leakage through Power Automate flows and canvas apps.
- **Cross-tenant inbound and outbound restrictions:** Use cross-tenant inbound and outbound restrictions to prevent the use of foreign public cloud services by your own users or your own public cloud services by foreign users.
- **Environment roles:** Use environment roles to manage proper access to environments and environment permissions.
- **Canvas apps sharing:** Share canvas apps properly within the organization.
- **Power Automate flows sharing:** Share Power Automate Flows properly within the organization.
- **Connector permissions:** Ensure proper use of connector permissions to provide every user access to only the necessary systems and data.
- **Model-driven apps access authorization:** Use model-driven apps access authorization to provide every user group within the organization access to only the necessary model-driven apps.
- **User accounts provisioning governance:** Use user accounts provisioning governance to avoid provisioning user accounts into unwanted Dataverse environments.
- **Session governance:** Use session governance to prevent the misuse of Dataverse applications by unauthorized users and to enforce policy changes that are applied during the user authentication process.
- **Service authentication:** Use the proper service authentication model when developing external applications that connect to Dataverse or Power BI environments.

- **Dataverse authorization:** Plan, design, and implement a proper Dataverse authorization model to ensure every user group has adequate permissions within Dataverse applications.
- **Portal authentication:** Choose proper portal authentication options to enable external users to authenticate with their favorite identity providers.
- **Portal authorization:** Plan, design, and implement a proper portal authorization model to ensure every external portal user group has adequate permissions within the portals.
- **Power BI authorization:** Implement proper Power BI authorization to enable access to all necessary Power BI components for every user group.
- **Row-level security:** Use row-level security within Power BI to manage access to Power BI data for different user groups based on role permissions.
- **DirectQuery security:** Use DirectQuery security to apply the authorization model of the underlying data source within Power BI.

Now that we've discussed all the security-related best practices and learned about establishing a mature security model for a Power Platform solution, let's go back to our fictitious customer, Contoso Inc., and see how they are going to implement authentication and authorization in their planned solution.

## Contoso Inc. security architecture

After a series of security workshops with their implementation partner Proseware Inc., and after gaining a full understanding of the Power Platform security possibilities, Contoso Inc. has created a security architecture for their Power Platform solution.

In this section, we will describe their security decisions in more detail.

## Active Directory integration

After Contoso Inc. already decided to use a two-tenant architecture, it was further decided that a **federation-based** integration will be implemented with their two AAD tenants. For this purpose, Contoso Inc. will establish a testing Active Directory forest and implement the **Azure AD Connect** component for both tenants with all features, including ADFS. This approach will allow them to keep full control over the user identities, security policies, and other already very well-established IT standards within their existing IT landscape. They will enable the following ADFS features:

- MFA
- Single sign-on

## Data Loss Prevention policies

To secure the Power Platform connector ecosystem, and in preparation for the future of enabling citizen developers within the organization, Contoso Inc. decided to make full use of the **Data Loss Prevention (DLP)** policies at the tenant level. This allows them to fully block some social connectors and include only the business-relevant connectors in the Business Data group.

## Dataverse

For Dataverse, model-driven apps, and Dynamics 365, Contoso Inc. decided on the following security principles:

- For every new Dataverse-based environment, a security/Office 365 group will always be created and assigned to ensure that no unwanted users are synchronized into the environment.
- Every new model-driven app will be assigned only the necessary Dataverse security roles, to avoid providing a high number of unnecessary apps to standard users.
- Session timeouts and inactivity timeouts will be made mandatory for every new Dataverse environment.
- For every new Dataverse environment, a proper authorization model will be configured, strictly using only the standard Dataverse authorization features. Using the record-sharing feature will be not permitted.
- Any integration with a Dataverse-based application will use only the OAuth authentication type with application users. Office 365 authentication or standard user-based OAuth authentication will be not permitted. The application user accounts in AAD will have separate security policies to reflect the non-interactive nature of these accounts, for example, no MFA.

## Other security decisions

Contoso Inc. will carefully consider the possibilities of the **ADFS Claims Rules** to implement conditional access, but right now, this feature will not be implemented. The employees of Contoso Inc. are working in many countries and are traveling for business purposes a lot, so those restrictions are not needed in the first place. Should Contoso Inc. observe malicious behavior and security breaches, this feature will be implemented immediately.

Contoso Inc. will seriously consider using the **DirectQuery** option for their **Power BI** solution against Dataverse, to leverage the integrated authorization capability of this approach.

Contoso Inc. will also consider using a third-party solution to replicate permissions from their Dataverse-based, model-driven, and Dynamics 365 solutions to the integrated SharePoint site collections that are used for document management.

After implementing the described Power Platform security architecture, Contoso Inc. is ready for the next steps in their implementation project.

## Summary

In this chapter, you learned a lot about Power Platform security, authentication, and authorization, as well as the different types of identities and the integration options available within an on-premises Active Directory. You also looked at the different ways to authenticate external users. After that, you analyzed the various authorization concepts in the Power Platform components and learned about **compliance, privacy, and data protection**.

With this knowledge, you should be able to define a security architecture for your own Power Platform solution, choose the best Active Directory integration, and benefit from the various **AAD** security options. Furthermore, you should be able to configure the authorization of all Power Platform components you plan to use and leverage for security best practices.

Now that we can design the security architecture of the solution, in the next chapter, we will focus on the client-side and server-side extensibility options of the Power Platform solution components, in particular **Dataverse**.



# 8

## Microsoft Power Platform Extensibility

In this chapter, you will learn about the customization and extensibility options available for Power Platform components. Power Platform components are designed to make it possible for you to build advanced solutions with configuration and customization only. However, certain requirements might dictate that we use custom development. It is important to understand the extensibility options of the Power Platform components in order to make the right decisions when designing a solution. The preferred way is to always use configuration and customization in the first place, before considering custom development, and to always follow the best practices for Power Platform extensibility.

In this chapter, we are going to cover the following main topics:

- Extensibility overview
- Dataverse and model-driven app extensibility
- Canvas apps and Power Automate extensibility
- Power BI extensibility
- Power Platform extensibility best practices
- Contoso Inc. Power Platform solution design

## Contoso Inc. – designing the Power Platform solution

Contoso Inc., with the support of their implementation partner Proseware Inc., was able to deeply analyze Power Platform's security possibilities. After understanding all the options, they designed a mature security model for their future Power Platform solution.

As the next step in their implementation project, Contoso Inc. needs to fully understand the possibilities of the Power Platform components in order to architect, design, and build a complex solution using configuration, customization, or custom development. They are interested in understanding, considering, and following the best practices surrounding extensibility.

### Getting an overview of extensibility

The ultimate goal of the Power Platform cloud service is to give a product to customers to build business applications quickly, easily, and with less custom development effort. Power Platform components offer a lot of business value already by default, by leveraging the out-of-the-box capabilities. In addition, all of those components offer a lot of extensibility options. There are four levels of extensibility for such components, as illustrated in the following diagram:

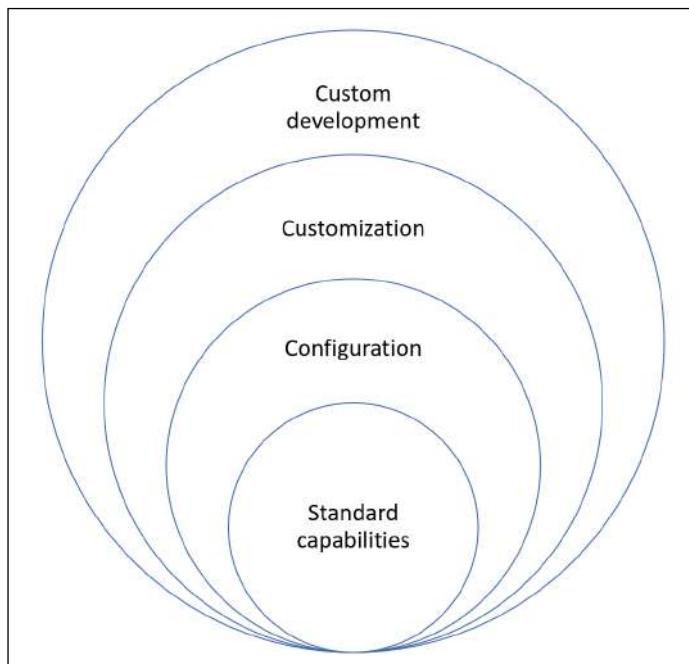


Figure 8.1: Power Platform extensibility options

As we can see, the options are as follows:

- **Standard capabilities:** Some of the Power Platform components can, theoretically, be used as they are, and this is true specifically for Dynamics 365 apps, which already implement business workloads. However, in most cases, it is necessary to extend the standard capabilities to cover specific customer requirements. Some other Power Platform components do not provide any ready-to-use capabilities, but rather a platform for building solution components. Enabling standard capabilities does not require any effort.
- **Configuration:** Configuration is the simplest extensibility option for Power Platform components. We define configuration as everything that can be done by using simple settings, entering configuration data, and so on, without modifying the structure of the component. A good example of configuration can be defining the product catalog in Dynamics 365 Sales or designing marketing emails in Dynamics 365 Marketing. Configuration can be, in most cases, performed by the administrators or Power users of the solution.
- **Customization:** Customization is the process of extending the capabilities of the Power Platform components by extending the structure or building new solution components. However, this is always done without using a programming language. A good example of customization can be extending the Dataverse data model and user interface, as well as building canvas apps, Power Automate flows, or Power BI datasets, reports, and dashboards. For this type of extensibility, a citizen developer's skills are sufficient, like those of business consultants or trained customer Power users.
- **Custom development:** Custom development is the most complex type of Power Platform extensibility. It always requires using a programming or scripting language to extend the solution capabilities. Good examples of custom development can be developing plug-ins or custom workflow actions in C#, client-side event handlers in JavaScript, or Power Apps Component Framework controls in TypeScript. Custom development is an area dedicated to IT pro developers, since deep IT knowledge is required.

In the following sections, you will learn about the extensibility options for the main Power Platform components, with a clear focus on the **customization** and **custom development** options. Let's start by looking at the extensibility of the Dataverse and model-driven apps.

## Presenting Dataverse and model-driven app extensibility

Dataverse is, from an extensibility point of view, the most complex Power Platform component. A lot of business capabilities can be achieved using the standard customization already, though there are many ways to fulfill complex business requirements using *automations* and *client-side* or *server-side custom development*.

In the following sections, we will explore standard Dataverse customization and automations, as well as possibilities for custom development, such as a client-side JavaScript event handler, client-side UI extensions with Power Apps Component Framework, or server-side extensibility with plug-ins.

### Dataverse standard customization

Dataverse standard customization encompasses a lot of different parts of the structure of Dataverse, offering us the ability to make various changes in order to implement specific customer requirements. Details about the different customization possibilities will be presented in the following sections of this chapter. However, it is necessary to generally understand the relationship between Dataverse and model-driven applications.

In every Power Platform environment, there can be only one Dataverse instance, and the whole customization is performed within this one instance of Dataverse. It is, however, possible to create multiple model-driven applications, and every such application can provide a subset of the structure of Dataverse to the end user so that the relationship between Dataverse and model-driven applications is *1:N*. The main purpose of this approach is to be able to build a consistent overall solution, but also offer every user group just the necessary subset of the features. This can avoid overwhelming the users with a plethora of unnecessary features and increase the overall adoption of the Dataverse solution.

The relationship between model-driven applications and Dataverse can be illustrated with the following diagram:

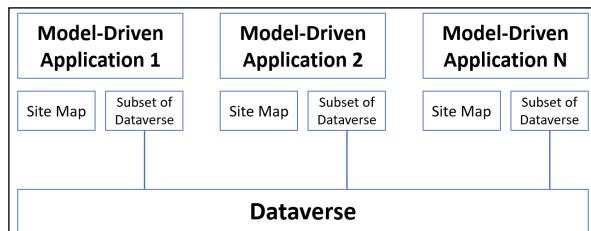


Figure 8.2: Relationship between model-driven apps and Dataverse

As we can see, every model-driven app consists of the following main parts:

- **Site Map:** The site map contains the main application navigation structure.
- **Subset of Dataverse:** A subset of Dataverse contains a collection of references to selected Dataverse artifacts like tables, forms, views, charts, etc.

Which artifacts can be selected, and which are included automatically, can be explained using the following simplified diagram:

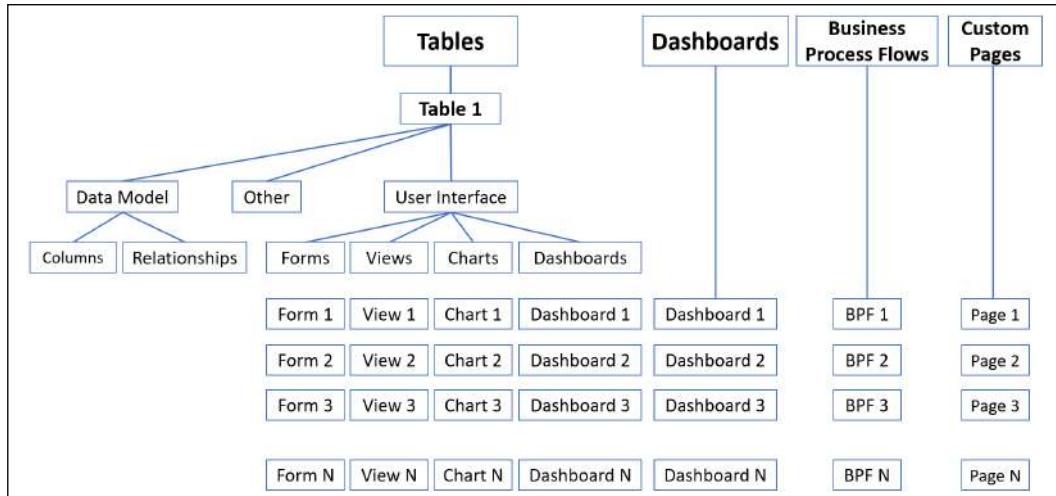


Figure 8.3: Selectable artifacts for model-driven apps

The preceding diagram presents a very simplified structure of the various Dataverse artifacts, just for the purpose of explaining the *model-driven apps* approach. When configuring a model-driven app, it is possible to select individual tables from the overall tables collection from Dataverse. When selected, the following artifacts of the table are automatically included in the app:

- The whole **data model** of the table, consisting of columns, relationships, and other artifacts
- **Other artifacts**, such as business rules, hierarchy settings, and alternate keys

There are also table artifacts, which can be individually selected as part of the model-driven app:

- **Forms:** It is possible to select one or more existing table forms.
- **Views:** It is possible to select one or more existing table views.
- **Charts:** It is possible to select some of the existing table charts. Table charts are not mandatory, so there is no requirement to include any charts in the model-driven app.

- **Dashboards:** It is possible to select some of the existing **table-specific** dashboards. Table dashboards are not mandatory, so there is no requirement to include any dashboards in the model-driven app.

Besides these table-specific artifacts, it is also possible to individually include the following table-independent artifacts:

- **Dashboards:** It is possible to select some of the existing **table-independent** dashboards. If dashboards are included in the model-driven app's site map as a navigational element, it is necessary to select at least one dashboard.
- **Business process flows:** It is possible to select some of the existing **business process flows (BPFs)**. There should be at least one BPF for every table that is BPF enabled and included in the model-driven app's site map.
- **Custom pages:** It is possible to either select an existing custom page or create a new one directly in the app designer. Custom pages are created using canvas apps technology.



#### Important note

It is necessary to include all tables used in the model-driven app's site map in the application itself.

The standard customization of Dataverse consists of certain basic areas:

- **Dataverse data modeling:** Used to design and implement the required data model
- **Dataverse user interface design:** Used to design and implement the standard user interface elements
- **Dataverse automations:** Used to design and implement standard Dataverse automations

The Dataverse standard customization is performed with the Maker Portal, as described in *Chapter 4, Power Platform Customization and Development Tools and Techniques*.

## Dataverse data modeling

Dataverse comes with a basic data model consisting of standard, general-purpose tables that are used in every relationship management solution, such as the Account table, representing the company customers, and the Contact table, representing the people to contact. The basic Dataverse data model can be extended in the following ways:

- Any **Dynamics 365 application**, such as Dynamics 365 Sales or Dynamics 365 Marketing, extends the data model with all workload-relevant tables.

- Any **third-party Dataverse-based application** coming from an implementation partner or from AppSource extends the data model with all required tables, columns, and relationships.
- **Customer-specific standard customization** performed within a solution implementation project can also extend the data model accordingly.

The following are some different ways to extend the Dataverse data model, regardless of the presence or absence of any Dynamics 365 or third-party Dataverse application already installed in the Dataverse environment:

- **Basic, standard, general-purpose Dataverse tables** cannot be deleted, and there are limited possibilities when it comes to modifying the standard table properties, columns, or relationships. It is possible to extend the data model with custom columns and relationships. These custom artifacts can be modified or deleted at any time.
- Any tables coming from a **Dynamics 365** or a **third-party Dataverse-based application** within managed solutions cannot be deleted. There are limited possibilities when it comes to modifying the standard table properties, columns, or relationships, as specified in the managed properties of these artifacts. It is possible to extend the data model with custom columns and relationships. These custom artifacts can be also modified or deleted at any time.
- It is possible for you to create your own **custom** tables with custom columns and relationships. These artifacts can be modified or deleted at any time, since you are the owner of those customizations.

When creating a new custom table, the following are the most important options to consider:

- **Table type:** A table can be a **standard** table, an **activity** table, or a **virtual** table. An activity table extends the existing group of activity table types (email, appointment, phone call, letter, fax, and so on) in Dataverse with a new activity type. The new custom activity table will be handled within the Dataverse solution's activity management like any other activity. A virtual table is a specific table type described later in this section.
- **Table ownership:** A table can be organization-owned or user/team-owned, as described in *Chapter 7, Microsoft Power Platform Security*. This setting has a significant impact on the authorization settings for this table.
- **Other settings:** You can also specifying some additional table behavior, such as the possibility for activity records to be assigned to the table in order to support emailing to the table and to enable connections, access teams, and document management.



### Important note

Certain table settings cannot be modified after the table has been created, so it is necessary to plan for those settings accordingly. Any incorrect settings would require the table to be deleted and completely recreated.

Now, let us describe the specifics of virtual tables.

## Virtual tables

**Virtual tables** are technologically different from standard or activity tables since they are not physically stored in Dataverse. The virtual tables capability makes it possible to include certain data from external databases into Dataverse to give the end users a seamless experience in the model-driven app, regardless of where the data is physically stored.

The creation of a virtual table first requires having a proper **data provider**, compatible with the **application programming interface (API)** of the external database. By default, there is an *OData 4* provider available in Dataverse; in addition, there is an *Azure Cosmos DB* provider available for free from Microsoft AppSource ([https://appsource.microsoft.com/en-US/product/dynamics-365/mscrm.documentdb\\_data\\_provider](https://appsource.microsoft.com/en-US/product/dynamics-365/mscrm.documentdb_data_provider)). If the external database has a specific API, not compatible with the above two data providers, a *custom provider* can be developed with code.

The next step is to create the virtual table, where the metadata will be physically stored in Dataverse, but the data itself will remain in the original database. Virtual tables have certain limitations, described below in the product documentation. However, they can be made part of a model-driven app, where they can be used in the usual way by the end users.

The virtual tables approach can greatly simplify certain data integration scenarios, as further explained in *Chapter 9, Microsoft Power Platform Integration*.



### Important note

For further details about virtual tables, please refer to the product documentation:  
<https://learn.microsoft.com/en-us/power-apps/maker/data-platform/create-edit-virtual-entities>.

When a **custom table** is created, table columns and relationships also need to be created. When creating new columns, the following are the most important options to consider:

- **Column type:** There are three basic column types: **standard**, **calculated**, and **rollup**. The column type cannot be changed after the column has been created. Values in the calculated and rollup columns are calculated automatically by the platform, and the columns are generally read-only.
- **Data type:** There are several standard and Dataverse-specific data types available. The data type cannot be changed after the column has been created.
- **Required status:** This setting defines whether a value in the column is mandatory when creating or modifying records in the table. There are three possible values for this setting: **Required**, **Recommended**, and **Optional**. Only the **Required** status type enforces entering data into the column upon creating a record.
- **Other column settings:** There are several other important settings, such as whether the column should be searchable, audited, and additional settings for interactive dashboard behavior.

When creating new relationships, the following are the most important options to consider:

- **Relationship type:** There are two main relationship types – **one-to-many/many-to-one** and **many-to-many**. They define basic relationship behavior. For many-to-many relationships, a hidden intersect table is created between the two related tables.
- **Type of behavior:** There are specific settings that define how the relationship will behave for the records on the *many* side of the relationship, when certain events happen on records on the *one* side of the relationship. The available configurable events are *Delete*, *Assign*, *Share*, *Unshare*, and *Reparent*. There are three basic settings: **Parental**, **Referential**, and **Custom**. Choosing the correct setting for the relationship behavior is of particular importance since it will influence the end user experience when working with data in a significant way. These types of decisions must be made during the design phase of the solution's implementation.

#### Important note



It is not in the scope of this book to dive deeper into the details of standard Dataverse data modeling. For further details, please refer to the product documentation: <https://docs.microsoft.com/en-us/powerapps/maker/common-data-service/data-platform-intro>.

In the next section, we are going to focus on how to design the Dataverse user interface elements.

## Dataverse user interface design

The visible part of the overall Dataverse solution is the user interface. The various user interface elements give the user the ability to see the processed data in a tabular or graphical representation. It also allows them to interact with the data by creating, modifying, and deleting records, or by triggering various functions.

In the following sections, you will learn the most important information you need to know about user interface elements.

### Table forms

Table forms are used within a model-driven app for creating, displaying, and modifying data in individual Dataverse table records. The following table form types are available:

- **Main Form:** The main form is used for creating, displaying, and modifying individual Dataverse table records. There can be multiple main forms per Dataverse table, and the main forms can be assigned to Dataverse security roles to be able to assign the best table form to the appropriate user groups. Furthermore, the main form can be deactivated so that it's unavailable for all users. Many different control types can be placed on the main form, including **iFrames**, **web resources**, **custom controls**, **PCF controls**, or **canvas apps**. These advanced possibilities will be described later in this chapter in more detail.
- **Quick View Form:** The quick view form is used for embedding onto main forms. Its purpose is to visualize certain data from related tables (from an *N:1* relationship) on the main form. The quick view form can contain columns from the related table, as well as subgrids displaying data from related tables of the quick view form's table itself. The data in the quick view form is always read-only when embedded in the main form. The quick view form can only contain a limited number of control types. There can be multiple quick view forms per table. Quick view forms are also used to design the hierarchy setting, as described later in this chapter.
- **Quick Create Form:** The quick create form can be used to quickly create records in a certain table by entering data into the most important columns. There can be multiple quick create forms per table, but only the first one created will be used; the additional forms are ignored. The quick create capability must be separately enabled for the respective table. The quick create form can only contain a limited number of control types.
- **Card Form:** The card form is a specific form type used to design compact views for mobile devices and streams for interactive experience dashboards. The form must have a specific design and can only contain a limited number of control types.

The following screenshot is an example of a main form showing an Opportunity record (part of Microsoft Dynamics 365 Sales):

The screenshot shows the Microsoft Dynamics 365 Sales Opportunity main form. At the top, there's a command bar with standard navigation icons like Save, Refresh, and Export to PDF. Below the command bar is a header section showing the record title "1 Café Grande Espresso Machine for Alpine Ski House - Saved", the status "Opportunity", the estimated close date "09.12.2022", the estimated revenue "14900,00 \$", and the owner "Robert Rybaric". A circular progress bar at the top indicates the current step is "Qualify (28 D)".

The main body of the form is divided into several sections:

- Summary:** Contains basic information like Topic (1 Café Grande Espresso Machine for Alpine...), Contact (Cacilia Viera), Account (Alpine Ski House), Purchase Timeframe (Unknown), Currency (US Dollar), Budget Amount, Purchase Process, Forecast Category, and Pipeline.
- Opportunity Sales Process:** Shows the current step is "Qualify (28 D)" and lists the steps: Qualify, Develop, Propose, Close.
- Sequence:** Opportunity nurturing, Step 1: Introduction mail (Email, Mark complete).
- Timeline:** Shows a search bar for the timeline and an auto-post field.
- Assistant:** Notifications: 1 reminder, Opportunity closing soon (1 Café Grande Espresso Machine for Alpine Ski House). Stakeholders: Cacilia Viera, Stakeholder (CV); Dwayne Elijah, Stakeholder (DE). Sales team: Jeremy Johnson, Sales Professional (JJ).

Figure 8.4: Example of a main form

As you can see in the screenshot, a main form of a table consists of several parts; you can see the command bar and the header in the upper part of the form, and below the header, there is a business process flow. Below that, you can see the body of the form consisting of tabs and sections, with content on every section.

The next type of user interface element we will discuss is the table view.

## Table views

Table views are used within a model-driven app to display and also modify table records.

There are the following view types:

- **System views:** In every Dataverse table, there are always the following system views created: *Advanced Find View*, *Associated View*, *Lookup View*, and *Quick Find View*. These views are used in model-driven apps for certain standard purposes, like presenting the result of search tasks, etc. The views cannot be manually created or deleted from a table, only customized in content.
- **Public views:** There can be multiple different public views created in a table. The public views are available in model-driven apps directly to the users to be selected for viewing or modifying data in tables.
- **Personal views:** These view types can be created directly by the end users, if they are authorized to do that. A personal view is available only to the creator, or when shared, to the users with whom it was shared.

The following parameters can be configured when creating new, or modifying existing, table views:

- Selecting table columns and defining their position and width
- Sorting records across multiple columns
- Filtering records using a comprehensive filter editor

It is possible to configure a Dataverse table so that it supports alternate view types. Examples of such alternate visual representations of table records are as follows:

- **Editable views:** This option makes it possible to update table records directly in the table view, without the need to open the record in a table form.
- **Calendar views:** This option makes it possible to display the table records on a calendar control. This option is suitable for tables that represent time entries with typical columns, such as *Time from* and *Time to*.
- Any other standard or custom-developed custom controls.

In the following screenshot, there is an example of a table view:

Topic	Potential Customer	Email Address	Status	Actual Close Date	Actual Revenue	Est. Close Date
1 Cafe BG-1 Grinder for Alpine...	Alpine Ski House	dwayne@alpineski...	Open		05.12.2022	
1 Café Grande Espresso Mach...	Alpine Ski House	cassandra@alpineski...	Open		09.12.2022	
10 Airport XL Coffe Makers f...	Alpine Ski House	cassandra@alpineski...	Open		16.12.2022	
10 Café A-100 Automatic Esp...	Fabrikam, Inc.	zoltan@fabrikam...	Won	06.11.2022	96500,00 \$	06.11.2022
12 Café A-100 Automatic Esp...	Fabrikam, Inc.	zoltan@fabrikam...	Won	01.11.2022	114000,00 \$	01.11.2022
18 Airport Coffee Makers for...	Northwind Traders	miguel@northwi...	Open		07.12.2022	
2 Café Corto for Northwind Tr...	Northwind Traders	miguel@northwi...	Open		31.12.2022	
2 Café Duo Espresso Machine...	Fabrikam, Inc.	zoltan@fabrikam...	Open		08.12.2022	
2 Semiautomatic Espresso M...	A. Datum Corpor...	keving@adatum.co...	Open		26.11.2022	
20 Café A-100 Automatic	Trey Research	alex@treyresearc...	Won	28.10.2022	167000,00 \$	28.10.2022
1 - 105 of 105						

Figure 8.5: Example of a view

As you can see in the example, a view has its own command bar with the buttons necessary for triggering the various transactions, and below the command bar, there is the view selector for selecting any of the available public views. In the main part of the view, there is a grid with the records from the selected table.

Table views, compared to table forms, cannot be assigned to security roles, so there is no way to configure user-specific table view assignments. Public views, however, can be deactivated so that they're unavailable to all users.

Next, we'll discuss table charts.

## Table charts

Table charts are graphical representations of data in a certain Dataverse table. Table charts can be visualized either as part of table views or by placing them on Dataverse dashboards. Charts are generally interactive, so it is possible to perform a drill-down into the data by clicking on some aggregates in the chart. When creating table charts, the following configuration possibilities are available to you:

- A table chart is always table-specific, so it can only visualize data from a single table.
- There are different selectable types of charts, for example, column, bar, area, line, pie, funnel, tag, and doughnut.
- Column series for values (*y-axis*) with different aggregations (sum, average, count, min, and max) and column categories for categories (*x-axis*) can be selected.

- It is possible to specify top-x or bottom-x rules to limit the number of displayed values on the chart.

In the following screenshot, there is an example of a table chart:

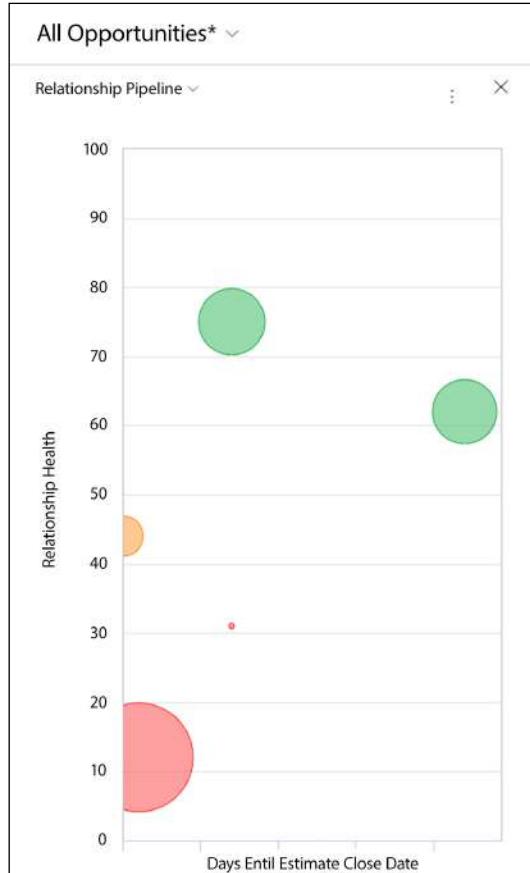


Figure 8.6: Example of a chart

As you can see in the screenshot, a chart is a graphical representation of the data from a table; in the above case, it is a specific chart for the Opportunity data.

Table charts have similar behavior to table views as they cannot be assigned to security roles, and there is also a possibility for end users to create their own **personal charts**.

## Table-specific and table-independent dashboards

Dashboards are user interface elements that display a collection of individual items. The following types of dashboards are available:

- **Standard dashboard:** This type of dashboard can contain table views, table charts, web resources, iFrames, and special components such as timelines, sales assistants, or organization insights. The components of the standard dashboard are independent of each other and do not automatically refresh.
- **Interactive experience dashboard:** This type of dashboard can be used directly as a workplace for a specific user role, since its components allow for various actions to be taken on the displayed data. The components are refreshed automatically and are mutually synchronized. This type of dashboard can contain table streams, and the data is filtered using visual filters.
- **Power BI embedded reports or dashboards:** This type of dashboard is created as an embedded Power BI report or dashboard. It must be ensured that the expected group of users has access to the Power BI elements embedded in the model-driven app. In order to ensure access to the same data as in Dataverse, the Power BI dataset behind the dashboard or report should ideally use the *DirectQuery* mode.

In the following screenshot, you can see an example of a standard dashboard, contained in Microsoft Dynamics 365 Sales:

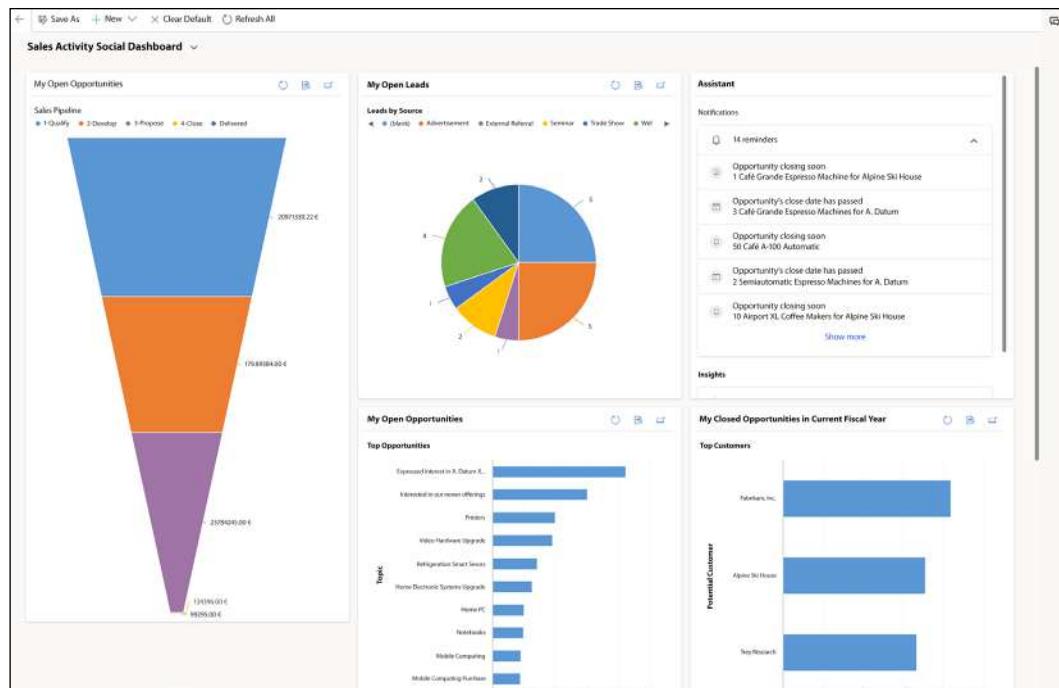


Figure 8.7: Example of a dashboard

As you can see in the screenshot, a standard dashboard can contain a chart, but also special tiles, like the **Assistant** tile.

Table-independent dashboards are used as home screens in model-driven applications, while table-specific dashboards are usually configured to be displayed instead of table views or within table forms.

Besides publicly available dashboards, users can create, save, and eventually share their own personal dashboards using the *dashboard designer* within a model-driven application. Dashboards can be assigned to security roles to limit dashboard availability to various user groups.

Personal dashboards have certain specifics compared to system dashboards. A user creating a personal dashboard can, for example, include Power BI tiles from Power BI dashboards when they have access to such dashboards. Furthermore, users can also include whole Power BI dashboards and save them as personal dashboards in Dataverse.

#### Important note



It is not in the scope of this book to dive deeper into the details of the standard Dataverse user interface design. For further details, please refer to the product documentation: <https://docs.microsoft.com/en-us/powerapps/maker/model-driven-apps/model-driven-app-components>.

## Other Dataverse customization artifacts

There are some other Dataverse standard customization artifacts worth mentioning:

- **Hierarchy Settings:** This feature makes it possible to visualize records in a certain table as a graphical hierarchy. In order to achieve this, it is necessary to perform specific configurations. You need to create or use a self-relationship within the table, create a specific quick view form for visualizing the most important data, and configure the hierarchy setting within a standard Dataverse customization.
- **Global choices:** **Choice** is one of the specific Dataverse data types. There are two types of choices – local choices, which are created directly within a table, and global choices, which are created outside of a table. Global choices can be reused in any table for a column that's of the **Choice** type. The benefit of global choices is having consistency in values of the same type across multiple tables.

- **Reports:** For years, the traditional reporting capability for Dataverse was the Microsoft SQL Server Reporting Services technology. This capability is still used, and every Dataverse-based solution usually comes equipped with a set of standard reports. Simple reports can be created directly with a wizard from within the Dataverse environment. However, for complex reports, it is necessary to use a *custom development* approach. Generally, traditional reports are losing relevance within Dataverse and are being replaced with more modern technologies such as Power BI.

Now that you have an understanding of all the main components of a Dataverse application, let's summarize the process of designing a model-driven app.

## Designing model-driven applications

As explained in the introduction to this section, a model-driven app is just a reference that provides access to certain selected parts of the overall Dataverse solution to end users. The configuration of the app is performed in the **app designer**, which is part of the Power App maker portal. The steps for creating a model-driven app are as follows:

1. Have all the necessary Dataverse solution artifacts ready.
2. Create a new model-driven application, give it a name, (optionally) a custom app tile, and a few other settings.
3. Configure the app's site map to provide the required navigation for the app. All the tables that are selected in the site map are automatically added to the app's configuration.
4. Select the app's artifacts (table-independent dashboards; business process flows; tables with selected forms, views, and charts; and table-specific dashboards) that should be included in the app.
5. Save and publish the app.
6. Assign proper Dataverse security roles to the app in order to make the app available to the respective user groups.

A model-driven app and a corresponding site map are technically two configuration files and, as such, can be part of a Dataverse solution package and can be deployed within the package to other environments.

## Designing mobile apps

Model-driven apps can be used on mobile devices (smartphones and tablets) when using the model-driven app/Dynamics 365 app. We can configure certain components of a model-driven app differently for mobile devices compared to PCs:

- On the table main forms, the tabs, sections, individual columns, subgrids, or other components can be made hidden on phones.
- Dashboards can be made hidden on phones.

Apart from these specific settings, the structure of the model-driven app, as well as the user interface, is the same on mobile devices as it is on PCs.

## Dataverse automation

Dataverse supports multiple automation possibilities to simplify and consolidate certain processes, make data entry easier, and ensure that certain processes happen automatically. All the standard automations are sometimes referenced as **processes**. In the following sections, we will look into various automation approaches.

## Dataverse business rules

**Dataverse business rules** are the simplest automation option within Dataverse. The purpose of business rules is to provide certain automations on the client side or on the server side within a single Dataverse record. A business rule consists of a set of conditions and actions, as illustrated in the following diagram:

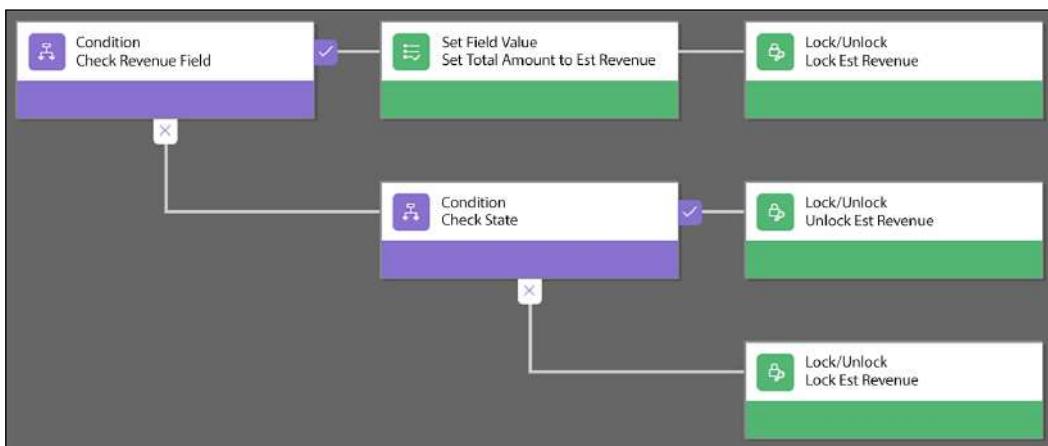


Figure 8.8: Dataverse business rule example

As we can see, a business rule can have multiple conditions and actions. The actions can be of the following types:

- Set the default value of a column
- Set the value of a column
- Lock/unlock a column on the table form
- Make the column visible/invisible on the table form
- Set the column as **Required** or **Optional** on the table form
- Show an error message on a column, based on a condition
- Provide a recommendation based on a condition

Business rules are created within the scope of a certain table, and all conditions and actions always refer to the values of a record in that table.

## Classic Dataverse workflows

**Classic Dataverse workflows** is a traditional automation approach that comes from the early days of Microsoft Dynamics CRM. The purpose of workflows is to perform certain background automations within Dataverse in a synchronous or asynchronous way.

### Important note



A synchronous workflow is executed as part of the main transaction (a new record is saved, an updated record is saved, or a record is deleted). This means that the workflow execution blocks the end user from continuing their usual interactive work with the model-driven app. The result of the workflow execution is visible immediately.

An asynchronous workflow is executed separately from the main transaction and does not influence the interactive work of the end user in any way. The result of the workflow execution is visible after a certain period of time (seconds to minutes after the main transaction is finished).

A Classic Dataverse workflow can be triggered by an event coming from a Dataverse record (record created, updated, or deleted), can be triggered from within a BPF, or can be triggered on-demand by a Dataverse user. A Dataverse workflow can implement business logic consisting of the following options:

- **Branching logic** (check condition, conditional branch, wait conditions, parallel waits, stop workflow, start child workflows, or custom actions)

- Dataverse record **transactions** (create record and update record)
- Trigger a **custom workflow action** (a type of custom-developed business logic, described later in this chapter)

#### Important note



The possibilities that are available when using traditional Dataverse workflows are limited compared to the more modern Power Automate flows. It is recommended to always prefer Power Automate, unless the automation must be implemented in a synchronous way, since Power Automate is asynchronous in nature.

Dataverse custom actions are very similar to Dataverse workflows. We'll look at these in the next section.

## Dataverse custom actions

The primary purpose of Dataverse **custom actions** is to provide an easily configurable piece of automation functionality that can be called from different places of a Dataverse application.

Dataverse custom actions are similar to Classic Dataverse workflows but have the following differences:

- Dataverse custom actions **do not have triggers**; they must be triggered from outside by Dataverse workflows, business process flows, or by code from within the Dataverse API.
- Dataverse custom actions allow us to define **input and output parameters** so that we can exchange values with the calling service.

Except for these differences, Dataverse custom actions provide the same business logic capabilities as Dataverse workflows.

Custom actions have a certain specific impact on the Dataverse API and the event handling mechanism. Upon creation of a custom action, there will be a **new API method** created automatically, as well as a **new event handling message**, which can be used for registering a plug-in, or another server-side event handler. This capability will be described in more detail later in this chapter.

## Dataverse custom APIs

A modern alternative to Dataverse custom actions is the **custom API capability**. A custom API can be created using one of the following ways:

- The Plug-in registration tool
- The Maker Portal
- With code
- Modifying the solution files

Using any of the methods, it is necessary to create a **custom API**, the required number of **request** and **response parameters**. If there should be a particular logic implemented for the custom API, a plug-in can be developed by code and registered for the custom API message in the event handling mechanism or Dataverse. A custom API creates a new API method and event handling message in the same way as a custom action described in the previous section.

### Important note



Please refer to the product documentation for more details about custom APIs:  
<https://learn.microsoft.com/en-us/power-apps/developer/data-platform/custom-api>.

## Dataverse business process flows

Dataverse BPFs are a different type of automation since these flows are not running in the background, but rather provide visual guidance to the end user when it comes to following certain business process. BPFs are also considered long-running automations since the process can remain in a certain stage for a very long time. An example of the BPF designer is illustrated in the following screenshot:

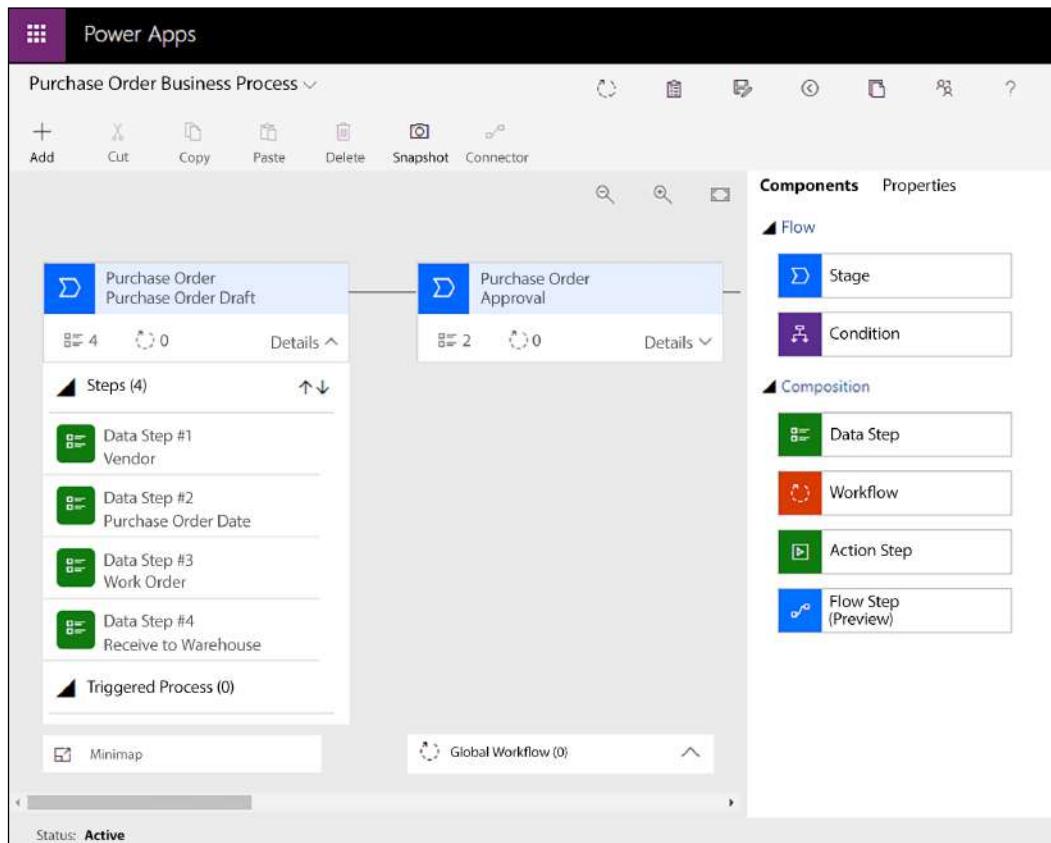


Figure 8.9: Business process flow example in the designer

As we can see, business process flows provide the following capabilities:

- A series of **stages** and **conditions** that guide the user through the main stages of a business process. These conditions can support implementing branching logic.
- In every stage, there can be a number of **data steps** for data entry, **action steps**, or **flow steps**, which can be triggered manually by the user in the respective stage to start a business logic process.

- In every stage, there can also be **workflows**, which will be triggered automatically on stage entry or stage exit.
- In addition, we can configure **global**, stage-independent workflows, which can be triggered during major business process flow events, such as when the BPF is applied or when the process is completed.

In Microsoft Dynamics 365 applications, there are always certain out-of-the-box BPFs that can be directly used or modified according to a customer's requirements. BPFs have the possibility to be connected with security roles, to limit the number of BPFs available to certain user groups. A BPF can encompass multiple tables, and there can be up to 10 different BPFs assigned in parallel to every Dataverse record.

With this, we have concluded the customization possibilities for Dataverse applications. In the next section, we will turn our attention to the extensibility of Dataverse, starting with client-side extensibility.

## **Dataverse client-side extensibility**

In this section, you will learn about the various extensibility options that are available on the client side of a Dataverse-based solution.

Dataverse offers a standard user interface design that is able to fulfill most of the usual customer requirements. In the case of specific and advanced requirements, there are multiple possibilities for improving the user interface's look and feel, as well as functional capabilities. You can also build completely customized client-side capabilities.

### **Standard custom controls**

The simplest way to improve the user interface of Dataverse, specifically table forms or views, is to use custom controls instead of the default controls. Dataverse provides a certain number of standard custom controls as part of the platform, and it is very easy to replace the default controls with custom controls. You can also develop your own custom controls and use them in the same way. This possibility will be described later in this chapter.

Different column-related standard custom controls are available for various column data types. These let you reflect the nature of the data type and provide a visually appealing and easy-to-use representation of the data. Replacing the default control can be performed within the table form designer in the Maker Portal. For every eligible column data type, you can select some of the available custom controls and decide what device type (web, tablet, or phone) this custom control will be used on.

The customization possibility is illustrated in the following screenshot:

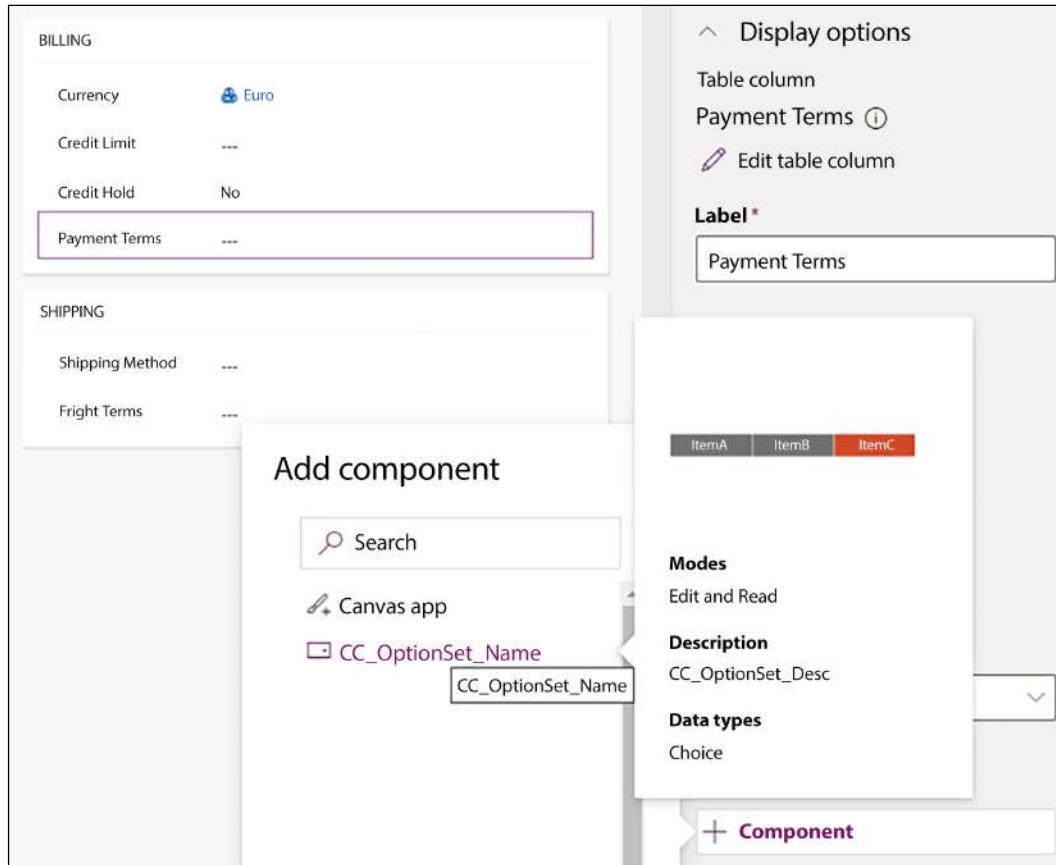


Figure 8.10: Configuring custom controls

As you can see, for a column of data type **Choice**, there is a built-in custom control that can be used on the table form instead of a default combo box style. The maker portal offers all available custom controls for the selected column and presents a preview of the look and feel of the custom control. This lets you easily select the most suitable control.

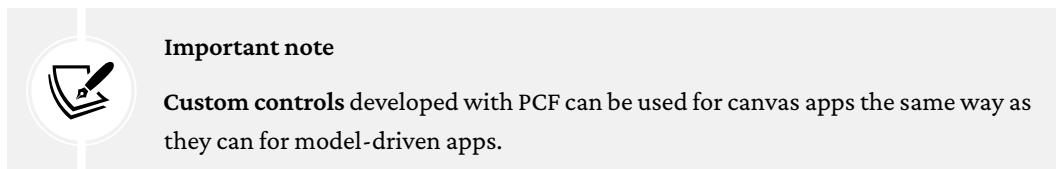
For certain custom controls, there might be the need for additional configuration settings, such as specifying the minimal and maximal values. If a table column visual representation is replaced with a custom control, the same visual representation will be used within business process flows in the data steps.

Standard custom controls can be also used to replace the table views with other user interface elements. Standard table views show the table records in a read-only table or grid visual representation. By using custom controls, this visual representation can be changed. Typical examples of custom controls are the **editable view**, **calendar view timeline control**, and **Kanban control**.

As for table columns, not every custom control can be used for every table. The respective table must have certain specific columns (for example, start and end date) and every custom control requires specific configuration settings. It is possible to enable multiple custom controls for an table, which gives the user the ability to switch the visual representation.

## Power Apps Component Framework

Power Platform allows users to develop their own custom controls to be used on the Dataverse user interface (table columns and views). The technology that's used for developing custom controls is called **Power Apps Component Framework (PCF)**, and the generic name for the developed components is **code components**.



PCF code components are solution-aware, so they can be included as part of a solution and deployed to other environments. The technical implementation of PCF code components consists of the following main elements:

- **Manifest:** The manifest is an XML file that specifies the structure of the PCF code component solution. All files used in the code component solution must be referenced in the manifest.
- **Implementation:** The PCF code components are implemented using the *TypeScript* scripting language. The main TypeScript file must have the name `index.ts` and must follow a certain prescribed structure.
- **Resources:** All additional files needed for the implementation of the visual part of the PCF code component (HTML, CSS, and others) are called **resources**.

As described in *Chapter 4, Power Platform Customization and Development Tools and Techniques*, the development of PCF code components requires the use of the **Power Apps command-line interface (CLI)** and a development environment such as **Visual Studio Code**. Since PCF code components haven't been integrated into Visual Studio yet, you need to use CLI commands for the development process. The overall process of creating a PCF code component consists of the following main steps:

1. Generate a PCF project structure using the Power Platform CLI `pac pcf init` command.
2. Retrieve project dependencies using the `npm install` command (**Node Package Manager (NPM)** is a tool that's installed along with the Power Platform CLI).
3. Modify the generated project files and, if needed, add additional project files using a code editor such as Visual Studio Code to implement the required capability of the component.
4. Build the PCF code component using the `npm run build` command.

After the PCF code component has been built, it is necessary to upload the component into Dataverse for use within Dataverse and for deployment purposes. This task also consists of several command-line commands. When the component is uploaded into Dataverse, it can be used for the table columns or views in the same way as standard custom controls.

#### Important note



For details about the Power Platform CLI, please refer to the product documentation: <https://learn.microsoft.com/en-us/power-platform/developer/cli/reference/>.

The benefits of using PCF code components to extend the Dataverse user interface are as follows:

- The PCF code components are part of the application rendering process, so their performance is optimized.
- It is possible for the code components to interact with the Dataverse API using the client-side Web API methods, without the need to specifically authenticate – the requests are automatically authenticated with the credentials of the current Dataverse user.
- It is possible to use device-specific features such as a camera, microphone, and GPS location. This capability can be used on mobile devices running model-driven apps to take pictures or sound recordings and add them to Dataverse records or to use the geolocation to add the GPS coordinates or address data to Dataverse records.

- The PCF components can be easily reused across a whole Dataverse solution.



### Important note

For further details and example PCF code components, please refer to the product documentation: <https://docs.microsoft.com/en-us/powerapps/developer/component-framework/overview>.

In the following sections, you will learn about the capabilities of Dataverse web resources.

## Web resources

Using web resources is the traditional approach for extending the Dataverse user interface, and it is a capability from the early days of Dynamics CRM. Web resources are still heavily used, though, and there are areas where web resources cannot be replaced with any other approach. Some typical usage scenarios for web resources are as follows:

- Building **web applications** or **controls** that will be embedded into the Dataverse user interface (table forms and dashboards).
- Implementing **client-side event handling** business logic.
- Implementing additional functionality for the local navigation – **extending command bars/ribbons**.

In the next section, you will learn more about the capabilities of web resources.

## Web resources overview

Web resources in Dataverse are basically different file types that are uploaded into the Dataverse database and used for different extensibility purposes. The web resources are solution-aware and can be deployed to other Power Platform environments. The following types of web resources exist:

- **User interface web resources:** These web resource types are primarily used to build additional user interface elements for Dataverse. Typical representations are *HTML* and *CSS* components.
- **Data web resources:** These web resource types are used to store small amounts of static data and are usually used together with the user interface web resources. Typical representations are *XML* and *XSD* components.
- **Graphical web resources:** These web resource types are used to enhance the custom user interface or for other purposes directly within the Dataverse. Typical representations are *PNG*, *JPG*, *GIF*, *ICO*, and *SVG* image components.

- **Code web resources:** These web resource types are used to implement certain business logic within the custom or standard user interface in model-driven apps. Typical representations are *JavaScript* and *RESX* components.

There are three different usage scenarios for web resources, as described in the following sections.

## Embedding web components

The purpose of web resources used in this role is to enhance the Dataverse user interface with non-standard UI elements or even with whole additional embedded solutions.

Compared to custom controls, web resources are not used to replace the visual representation of Dataverse tables, columns, or views; rather, they are placed on the Dataverse application UI (table forms and dashboards) using a dedicated form control type – a web resource. Only **form-enabled** web resources (HTML and graphical web resources) can be added to the Dataverse table forms and dashboards since it is necessary for them to have a visual representation.

You can develop whole complex web applications using the supported web resource types and then place the main application file, typically the *HTML page*, onto Dataverse forms or dashboards. In order to achieve this, all the used files need to be uploaded as web resources into Dataverse. To keep consistency within the web application, the individual files need to be referenced properly, as per the following example:

```
<link rel="stylesheet" type="text/css" href="..../styles/contoso.css" />
```

As we can see, a CSS stylesheet web resource is being referenced by an HTML web resource using a **relative URL**.

The web application can use JavaScript files to refer to all the Dataverse table form elements and to communicate with the Dataverse API using a client API object model. You will learn more about the client API object model in the next section.

It is also possible to use JavaScript files to communicate with any third-party web service endpoints to implement the required functionality.

## Form scripting

Web resources of code type (JavaScript) can be used to implement **client-side event handling** extensions. With form scripting, you can manipulate the Dataverse user interface elements (within Dataverse table forms) based on client-side events. There are multiple events that can be used to trigger JavaScript event handlers. All these events are related to table forms and their controls. The following are the most important and frequently used:

- **Form OnLoad:** This event happens during the initial load of a Dataverse table form. The event handlers registered here can perform initial manipulations of the table form's content.
- **Form OnSave:** This event happens during the save process of the form's data. The event handler registered here can perform certain manipulations, depending on the last state of the Dataverse table form's controls, immediately before the form's data is saved.
- **Column OnChange:** This event happens after the user leaves a Dataverse table form's column, if the value of the column was changed. The event handler registered here can perform certain manipulations, depending on the current value of the particular column.

There are other events related to specific parts of a Dataverse table form, such as subgrids, iFrames, lookup controls, the knowledge base search control, and the table tabs.

For the actual manipulations, which can be implemented within the JavaScript event handlers, a specific client API object model can be used. This object model makes it possible to perform a wide variety of manipulations, such as showing and hiding parts of the Dataverse table form (tabs, sections, columns, and other controls), performing calculations with data in the table forms, performing navigational actions, and performing actions with the BPF on the current record. In addition, the object model makes it possible to call any Dataverse API method using a client-side Web API. There are many other possibilities, such as using some device-specific features on mobile devices (images, audio, video, and geolocation), and much more.



#### Important note

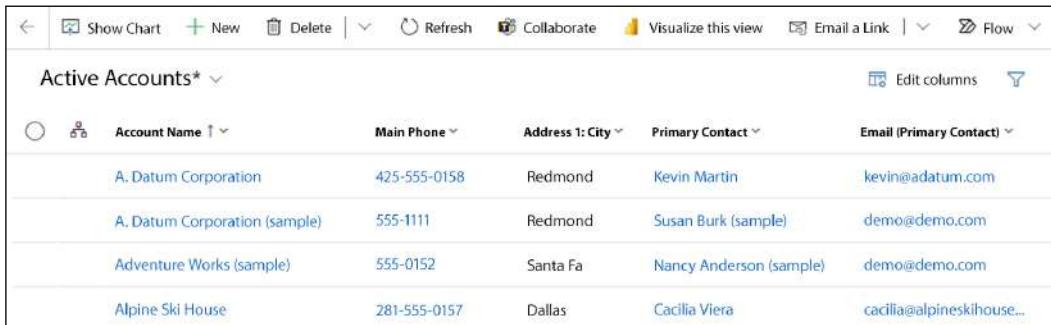
The whole client API object model is very complex, and it is not in the scope of this book to provide every detail about the API. For further details, please refer to the product documentation: <https://docs.microsoft.com/en-us/powerapps/developer/model-driven-apps/clientapi/reference>.

When comparing form scripting with Dataverse business rules, there is a certain overlap since the purpose of Dataverse business rules is to minimize the use of code. However, at the time of writing, the possibilities that are covered by Dataverse business rules are rather limited, covering only the most used types of client-side manipulations. There are still a lot of requirements that cannot be covered with Dataverse business rules and need to use form scripting.

## Command bar/ribbon extensibility

Web resources made with JavaScript code can be further used to extend the command bar or ribbon with additional features. Before we dive deeper into this type of client-side extensibility, it is important to understand what exactly command bars and ribbons are.

Besides the main navigation of model-driven apps, which is implemented as a **site map**, there is also a local navigation that's specific to every table type. This navigation is visually represented as the navigation bar with navigation elements (buttons) in the upper part of the table view, including forms and sub-grids, as illustrated in the following screenshot:



The screenshot shows a table view titled "Active Accounts\*". The command bar at the top includes buttons for "Show Chart", "New", "Delete", "Refresh", "Collaborate", "Visualize this view", "Email a Link", and "Flow". Below the command bar is a header row with columns for "Account Name", "Main Phone", "Address 1: City", "Primary Contact", and "Email (Primary Contact)". The data table contains five rows of account information:

Account Name	Main Phone	Address 1: City	Primary Contact	Email (Primary Contact)
A. Datum Corporation	425-555-0158	Redmond	Kevin Martin	kevin@adatum.com
A. Datum Corporation (sample)	555-1111	Redmond	Susan Burk (sample)	demo@demo.com
Adventure Works (sample)	555-0152	Santa Fa	Nancy Anderson (sample)	demo@demo.com
Alpine Ski House	281-555-0157	Dallas	Cacilia Viera	cacilia@alpineskihouse...

Figure 8.11: Command bar example

As we can see, there is a command bar in the table view of the Dataverse table **Account**. This provides various options within an table view such as showing a chart, creating a new record, or deleting a record. Command bars are table-specific. For every Dataverse table, the structure of the navigation elements can be different.

Within Dataverse, there are three table-specific command bars:

- **Main grid command bar:** This command bar presents all navigational options for a specific table when an table view with multiple records is displayed. The navigation elements can provide certain functionality to the **whole group** of displayed records.
- **Table form command bar:** This command bar presents all navigational options for a specific table when an table form with a single record is displayed.
- **Table sub grid command bar:** This command bar presents all navigational options for a specific table when a sub-grid of related records is displayed on an table form.

### Important note



In the legacy Dataverse table forms, as well as in the legacy Dynamics 365 Client for Outlook, the table-specific navigation was implemented differently, in the form of **ribbons**. Since the legacy user interface has been deprecated, it is not necessary to consider the ribbon extensibility for future Dataverse-based solutions.

Extending the command bar can be done to implement some of the following requirements:

- Reorder the standard navigation elements (buttons) on the command bar
- Hide some of the standard navigation elements (buttons) on the command bar
- Add new custom navigation elements (buttons) to the command bar, including a corresponding functionality

As explained in *Chapter 4, Power Platform Customization and Development Tools and Techniques*, the command bar can be extended either using the standard procedure by exporting the customization XML file, manually modifying the file and importing it back into Dataverse, or using a third-party tool called *Ribbon Workbench*. It is certainly recommended to use the latter options since they are much more convenient and do not lead to errors.

When adding new functionality to the command bar for a certain table, you have the following main options:

- Create a new navigation element (button) and assign a **URL** to the navigation element. When using this option, the resource that's specified in the URL is loaded when the button is clicked.
- Create a new navigation element (button), create a **JavaScript handler**, upload it as a web resource, and assign the **handler** to the navigation element. When using this option, the business logic that's implemented in the JavaScript code is executed when the button is clicked.

In both cases, the result for the end user will be a modified command bar with custom functionalities. When using the JavaScript option, the possibilities for implementing business logic are identical to the possibilities that are available when using form scripting.

### Important note



As of writing this book, there is a new low-code possibility to extend the command bar, natively incorporated into the Maker Portal. This capability makes it possible to graphically extend the command bar with new navigational elements (buttons and more) and implement the functionality behind the buttons either using the approach described above (URL or JavaScript handler) or using **Power Fx** formulas.

## Embedding canvas apps

Another interesting way to extend the Dataverse user interface is by embedding canvas apps into Dataverse table forms. This makes it possible to build tailored and visually appealing user interface elements. This customization is illustrated in the following screenshot:

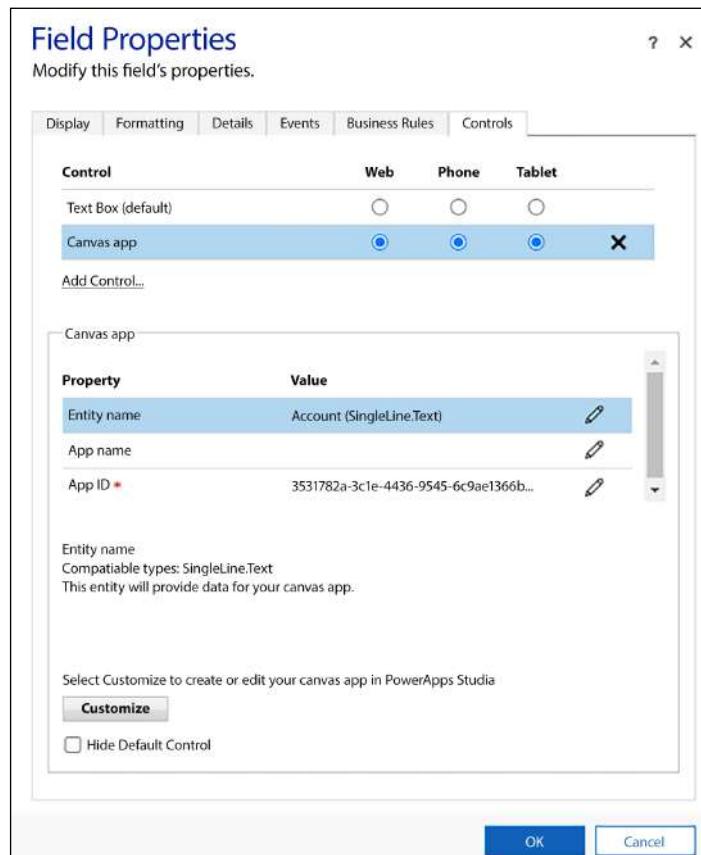


Figure 8.12: Embedding canvas apps

Embedding a canvas app into an table form consists of the following steps:

1. Select an table column. It is recommended to always use a column set to **Required** and place it onto a dedicated area on the table form (for example, an empty section).
2. Edit the properties of the column to add a new custom control to the column. In this case, it must be a custom control that's of the **Canvas app** type.
3. Select the **Customize** button to start the Power Apps Studio so that you can design the embedded canvas app.
4. Once the canvas app is ready, it needs to be saved to the cloud. This will generate a relationship between the Dataverse column and the canvas app so that the app will be displayed on the Dataverse table form instead of the Dataverse column.

#### Important note



It is not in the scope of this book to provide all the details about creating and embedding canvas apps into Dataverse table forms. Please refer to the following product documentation for further details: <https://docs.microsoft.com/en-us/powerapps/maker/model-driven-apps/embed-canvas-app-in-form>.

It is possible to implement any kind of user interface and business logic in the embedded canvas app by adding additional data connectors and any required UI controls. Since the embedded canvas app will live within an table form and should reflect the context of the current record, it is necessary to build the data structure that's used in the app around the primary table column that's used when creating the app.

Embedding canvas apps into Dataverse table forms currently has the following limitations:

- There can be only one enabled canvas app per table form.
- The creation and embedding process must follow the high-level steps described previously. You cannot embed a canvas app that's created outside of the described process.
- Canvas apps must be published separately. Publishing Dataverse customization does not publish canvas apps.
- Canvas apps must be shared with all Dataverse users separately. The Dataverse security roles do not give implicit permissions to view and use the embedded canvas app.

In the previous sections, you learned a lot about the different ways to extend the Dataverse user interface or implement certain client-side business logic. In the next section, we are going to focus on the server-side extensibility options within Dataverse.

## Dataverse server-side extensibility

Extending a Dataverse solution on the server side is an option that must be used mainly in the following scenarios:

- The Dataverse standard automation capabilities are unable to cover advanced requirements
- There is a need to develop a custom batch processing solution
- There is a need to develop an alternate client application
- There is a need to develop an incoming or outgoing integration with the Dataverse solution

In order to fulfill these advanced requirements, the Dataverse platform provides certain server-side extensibility options, all of which will be explained in more detail in the following sections.

## Dataverse API interface types

The foundation for any Dataverse server-side extensibility is the Dataverse API. The Dataverse API contains a rich set of interfaces that can be used to perform the **Create, Read, Update, and Delete (CRUD)** transactions on any data within Dataverse. It also allows you to call specific advanced methods going beyond the basic CRUD transactions. Dataverse offers the following types of API interfaces:

- **Web API:** This is the OData version 4 REST-based endpoint. It's recommended that you use it for all new custom development solutions. This endpoint provides full capabilities for all Dataverse transaction types.
- **Organization Service:** This is the legacy SOAP-based endpoint and is still used for certain custom development solutions (plug-ins, custom workflow actions, and so on). This endpoint provides full capabilities for all Dataverse transaction types.
- **Discovery Web Service:** This is a specific web service endpoint that's used to detect all Dataverse environments to which a particular user has access. This endpoint has a Web API, as well as a SOAP version.
- **Deployment Web Service:** This is a specific web service endpoint used for managing *on-premises Dataverse deployments only*.

- **Power Platform API:** This is a specific web service endpoint used for administering cloud Dataverse environments.
- **Tabular Data Stream (TDS):** This is the latest endpoint, and is used for read-only access to Dataverse data.

The recommendation about when to use which of the listed endpoints depends on the type of custom development that needs to be performed:

- Solutions that are implementing **plug-ins** and **custom workflow actions** must use the organization service with the respective assemblies, since this is the only supported way to perform actions against Dataverse.
- Solutions that are implementing **external** communication with Dataverse can use either the organization service or the Web API. Since, for an organization service, there is a collection of assemblies making the development process easier, this can be the best way to use an organization service when developing using .NET Framework. For solutions that are not using .NET Framework, the only way to communicate is to use the Web API. The recommended Microsoft assemblies to use for this type of development are the XRM Tooling assemblies described below.
- The deployment web service (for on-premise) or the Power Platform API (for the cloud) are used only for administration purposes.
- The **TDS** can currently only be used to **read data** from Dataverse, for example, within **Power BI**, using the **SQL Server Management Studio (SSMS)** or **SQL Server Integration Services (SSIS)**. It is also possible to use TDS in code to read Dataverse data and use it for processing.

All the solutions that implement external communication with the Dataverse API must contain proper authentication, as described in *Chapter 7, Microsoft Power Platform Security*. Plug-ins and custom workflow actions do not need to authenticate when communicating with the Dataverse API since they run in the context of the Dataverse platform, and the Dataverse API calls are pre-authenticated.

To support the custom development using some of the aforementioned APIs, Microsoft provides certain DLL assemblies as part of the NuGet developer tools and assemblies, as described in *Chapter 4, Power Platform Customization and Development Tools and Techniques*. The following is an overview of the most important assemblies from this package:

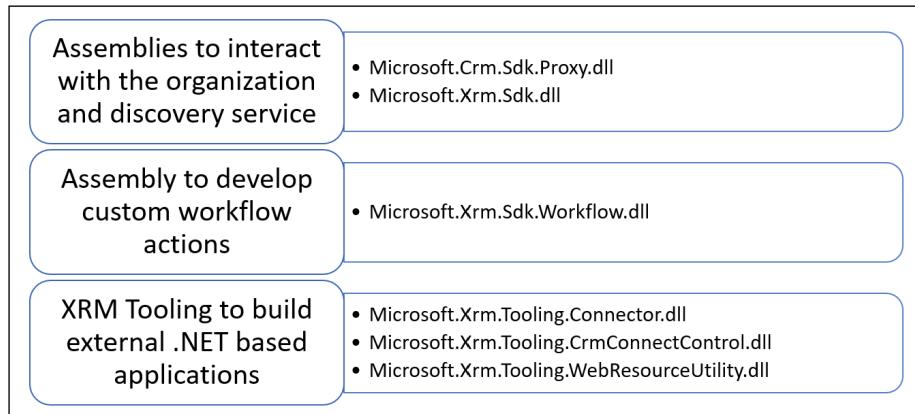
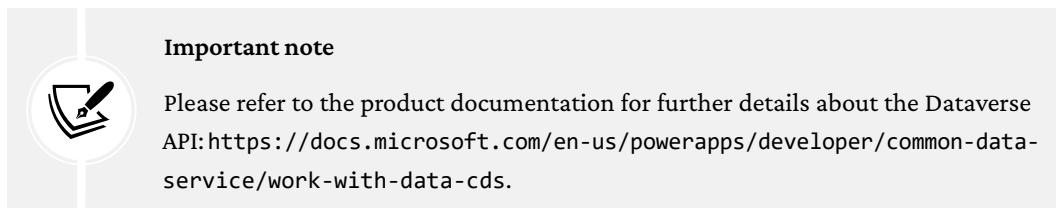


Figure 8.13: Dataverse developer assemblies

There are no Microsoft-provided assemblies available for Web API development. The developer needs to use the basic .NET assemblies for *HTTP* communication, such as `System.Net.Http`. However, there are already third-party NuGet packages with assemblies developed by community experts, which can be used instead of low-level HTTP communication. An example of such a package can be found in the following repository: <https://www.nuget.org/packages/Xrm.Tools.CRMWebAPI>.



In the following sections, you will learn about the various Dataverse server-side extensibility options.

## Plug-in event handlers

Plug-ins are server-side event handlers that work in a similar fashion as the Dataverse workflows. However, they must be developed using code. Plug-ins can be used to execute transactions against the Dataverse API or to communicate with external web service endpoints. Saying that, they can be used to develop advanced automation solutions or to implement outbound integration logic.

The process of developing and deploying plug-ins consists of the following steps:

1. Develop a plug-in event handler as a DLL assembly using Microsoft Visual Studio by following the mandatory plug-in code structure in a .NET-based language (preferably C#).
2. Sign and compile the DLL assembly.
3. Register the plug-in DLL assembly with Dataverse using the *Plug-in Registration Tool*, as described in *Chapter 4, Power Platform Customization and Development Tools and Techniques*.
4. Register the steps of the implemented event handler for the required events using the Plug-in Registration Tool.

After performing these high-level steps, the business logic that's implemented in the plug-in will start executing for all the registered events in the table. The following important settings are used while registering a plug-in step with the Plug-in Registration Tool:

- **Message:** This setting specifies the event or transaction that should fire the event handler's execution. There is a multitude of different transactions available, with many of them being table-specific. The most used generic transaction types are **Create**, **Update**, **Delete**, **Retrieve**, and **RetrieveMultiple**.
- **Primary table:** This setting specifies the table from which the records will trigger the plug-in's execution.
- **Execution context:** This setting specifies whether the event handler will execute in the context of a specified user or in the context of the calling user – the user who performed the transaction in Dataverse.
- **Pipeline Stage:** This setting specifies at which stage during the transaction the event handler will be executed. The possible values are **PreValidation**, **PreOperation**, and **Post-Operation**.
- **Execution Mode:** This setting specifies whether the event handler will execute synchronously as part of the main transaction or asynchronously as a separate post-transaction process. This setting defines the execution type exactly the same way as for Dataverse workflows.

### Important note



**Synchronous plug-ins** are executed as part of the execution pipeline, which means that the end user needs to wait until the whole execution is finished. When the plug-in execution takes a longer time to finish, the end user will be confronted with a frozen model-driven application. This is an unacceptable user experience. Therefore, it is required to use the synchronous mode only for plug-ins that are executing very fast.

It is also important to better understand the difference between the three possible pipeline stages:

- **PreValidation:** The event handler that's registered for this stage is executed at the beginning of the execution pipeline before the main transaction is executed. During this stage, the transaction can be canceled from within the event handler.
- **PreOperation:** The event handler that's registered for this stage is also executed at the beginning of the execution pipeline – after the *PreValidation* stage, but before the main transaction is executed. During this stage, the context data that's processed by the event handler can be modified before it is processed by the main transaction.
- **PostOperation:** The event handler that's registered for this stage is executed at the end of the execution pipeline after the main transaction has been performed. The context data cannot be modified anymore since the data is already physically stored in Dataverse. The business logic can implement additional transactions, such as creating, updating, or deleting records in other tables.

A plug-in can be used for the following main scenarios:

- **Implementing advanced Dataverse business logic:** In this scenario, the plug-in, which is triggered by some events in the Dataverse, implements certain specific business logic such as complex calculations over other Dataverse records.
- **Implementing outbound Dataverse integration:** In this scenario, the plug-in, which is triggered by some events in the Dataverse, implements an outbound integration. In other words, the plug-in sends data from the triggering Dataverse records to an external web service endpoint.

It is important to understand the limitations of using plug-ins with Dataverse, especially for calling external services:

- The plug-in's execution is limited to a general **2-minute timeout**. After this time, every running plug-in instance is automatically canceled.
- A plug-in can only call **HTTP-** or **HTTPS**-based external web service endpoints; no other external communication type is allowed. The endpoints must be of a DNS type; no direct IP addresses are allowed. There are no supported advanced authentication scenarios for calling the endpoints.

Due to these limitations, it might be sometimes necessary to consider a different technological approach in order to implement certain complex business requirements. An example of an alternate approach could be *Microsoft Azure* technology.

In the next section, you will learn how to extend the Dataverse workflows with advanced functionality using code.

## Custom workflow actions

Custom *workflow actions* (in contrast to *custom actions*) are code components that are developed in a similar way as plug-ins, but they are used for a different purpose. While plug-ins are event handlers that are triggered by Dataverse events, custom workflow actions are not event handlers and are not triggered automatically, but rather called as actions from Dataverse workflows. The process of developing and registering custom workflow actions is similar to plug-ins, except there is no registration of any step, only of the assembly itself. The code of a custom workflow assembly can contain input and output parameters, which are used to exchange values with the calling Dataverse workflow. The possibilities available when using the implemented business logic are exactly the same ones that are available for plug-ins; they can be used for implementing advanced automations or communication with external systems. Once registered, a custom workflow activity is made available from within the Dataverse workflow designer.

## Azure Service Bus integration

Dataverse, as a major component in Microsoft's cloud strategy, offers a variety of possibilities for integrating with other Microsoft cloud services. The purpose of these integrations is to provide users with an easy way to use additional extensibilities outside the Dataverse platform itself.

One of the standard integrations is between Dataverse and **Microsoft Azure Service Bus** in order to cover the following typical scenarios:

- **Remote code execution (relay)**: In this scenario, Dataverse events trigger the execution of a remote code component called the **listener**. This component receives the request, along with the **remote execution context** containing the data from the Dataverse record being triggered, and performs any necessary processing.

- **Outgoing integration:** In this scenario, Dataverse events send the remote execution context as **messages** via an Azure Service Bus queue or topic to a listener. The listener retrieves the messages from the queue or topic and can perform any necessary processing.

There are several technical differences between these two approaches, with the main difference being that, in the case of remote code execution, the listener component must run permanently to immediately respond to any execution request from Dataverse. For the queue- or topic-based scenario, the listener must not run permanently since the messages are waiting in the queue to be retrieved.

The solution architecture for Microsoft Azure Service Bus integration can be seen in the following diagram:

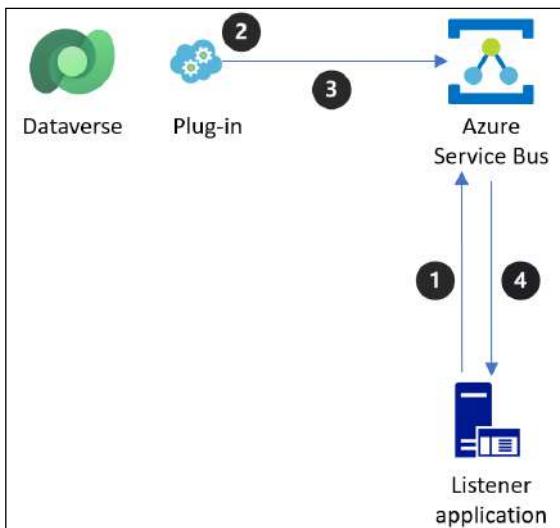


Figure 8.14: Dataverse Azure Service Bus integration

As you can see, the solution consists of:

- A **Listener application** (1), which runs either in the on-premises environment or in the cloud (for example, in an Azure VM). The listener application *opens an outbound* connection to the Azure Service Bus instance.
- A **built-in Dataverse plug-in** (2), which just needs to be configured to connect to the Azure Service Bus instance. After the connection and the triggering message(s) are configured, Dataverse starts to send the **remote execution context** (3) of the triggering record in case a registered event (for example, a record creation, update, or deletion) happens.
- The Listener application receives the **remote execution context** (4) for further processing.

The process of registering an Azure Service Bus service endpoint is illustrated in the following screenshot:

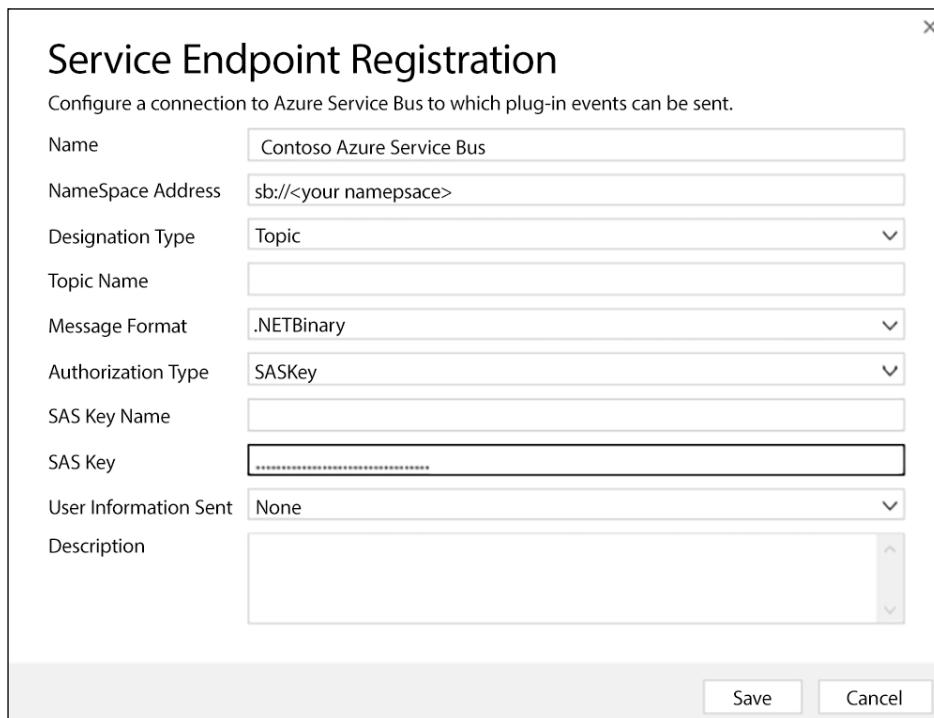


Figure 8.15: Azure Service Bus endpoint registration

When registering the service endpoint, it is required to configure:

- The endpoint's namespace
- The type (there are multiple values possible, specifying whether the communication will be *relay-based* or *message-based*)
- The message format (.NETBinary, JSON, or XML), which is important for the implementation of the Listener application
- The authentication settings

#### Important note



Due to the nature of this standard integration – the use of a built-in asynchronous plug-in – the solution always works end-to-end asynchronously, even for the relay patterns.

The biggest benefit and value proposition of this solution approach is that the connection between the cloud and on-premises is always outbound, from the data center into the cloud. This is much easier to configure since there is no need to allow inbound communication through the protective layers in the perimeter.

## Azure Event Hub integration

Dataverse also offers a very similar form of standard integration with Azure Event Hub, instead of Azure Service Bus. **Azure Event Hub** is an Azure component for building highly scalable publish-subscribe services. This type of integration requires an Azure Event Hub instance, instead of Azure Service Bus.

The registration on the Dataverse side is very similar, consisting of registering with Azure Event Hub and registering the execution step to trigger the integration. When configured, the Dataverse events will start sending the remote execution context as XML or JSON messages to Azure Event Hub. There is a need to implement the required business logic to process this data in the Azure environment.

## Webhook integration

Dataverse also offers built-in integration with a web service endpoint (**webhook**). This capability can, to some extent, replace the need to develop plug-ins for the purpose of outbound integration. In order to establish a webhook-based solution, the following steps must be performed:

1. A web service solution needs to be developed. This solution must provide the capability for an `HttpHeader`, `HttpQueryString`, or `WebhookKey` authentication and must accept the Dataverse remote execution context as a parameter. Regarding the usual Microsoft cloud technologies, Azure Functions or Azure App Services can be used to develop the web service.
2. A webhook and the necessary steps must be registered in the Dataverse environment for the required table and message type using the Plug-in Registration Tool.
3. A user performs a transaction in the Dataverse application that was configured for webhook integration.
4. The transaction triggers the integration, which forwards the remote execution context of the triggering Dataverse record to the web service for further processing.

In contrast to the Microsoft Azure Service Bus integration, webhook integration can be configured to run *asynchronously* but also in a true **synchronous** mode.

When deciding between Azure Service Bus and webhook, it is important to consider scalability. Since Azure Service Bus, especially when using queues or topics, is a messaging solution with large scalability, the webhook solution is only as scalable as the implementation of the web service permits.

## Building external applications

The last Dataverse server-side extensibility option is to build external applications and connect to the Dataverse API to communicate with Dataverse and trigger various Dataverse transactions. There are several scenarios where this approach can be used:

- Building batch jobs to perform mass processing solutions
- Building alternate desktop or web client applications communicating with Dataverse
- Connecting existing portal solutions from third parties with Dataverse
- Building mobile applications communicating with Dataverse
- Building integration scenarios to integrate between Dataverse and third-party IT systems with or without an integration middleware

In the following sections, you will learn more about some of these scenarios.

## Batch jobs

The standard extensibility patterns of Dataverse are mostly real-time or near real-time, like the event-based ecosystem described above.

For scenarios where event-based automation simply does not work, batch job processing might be necessary. As an example, the daily mass sending of birthday congratulatory emails to contacts stored in a Dataverse solution cannot be automated using any of the event-based approaches. When implementing Dataverse batch jobs, there are different ways to use standard technologies:

- **Power Automate:** Power Automate flows must not necessarily be triggered only by Dataverse events, but also by using a timer trigger. The timer-triggered flow can use all the capabilities of the Dataverse connector to get the required data and perform the necessary processing using the looping components.
- **Microsoft Azure Functions:** For more advanced scenarios that aren't covered by the capabilities of Power Automate, there is the possibility to use Azure Functions with a timer trigger. It is possible to implement any kind of processing complexity with code and communicate with the Dataverse using any of the preferred APIs.

There are certainly other possibilities when it comes to implementing batch jobs against the Dataverse API using on-premises solutions, but in this book, we will be discussing cloud-based solutions wherever possible.

## Alternate clients and portals

There might be situations where an alternate client, whether desktop or web, needs to be implemented to communicate and expose some of the data and functionality of Dataverse. In this case, the Dataverse API can be used to cover the communication with Dataverse, get or write the required data, or trigger processes. Depending on the technology, it can be either the SOAP-based interface for .NET-based solutions, or the Web API for everything else. As always, the OAuth-based authentication, where registration in Azure Active Directory is required, is the preferred way.

## Mobile clients

When developing a mobile solution for an **internal audience**, using the standard *Power Apps* or *Dynamics 365 mobile client* for model-driven or canvas apps is certainly the best option – there is no need to think about a different custom development approach. However, there is no standard technology as part of Power Platform that provides mobile applications for external public users.

Public mobile applications for iOS or Android can develop any required functionality and connect to the Dataverse Web API to perform the required communication. In order to authenticate against the API, it is necessary to again use *OAuth*. Microsoft provides certain supporting libraries for native mobile platform development to make it easier to implement OAuth against Azure Active Directory:

- **Microsoft Authentication Library (MSAL)** for Android: <https://github.com/AzureAD/microsoft-authentication-library-for-android>
- MSAL for iOS and macOS: <https://github.com/AzureAD/microsoft-authentication-library-for-objc>

### Important note



More details about Power Platform integration scenarios and solutions will be provided in *Chapter 9, Microsoft Power Platform Integration*.

In the next two sections, you will learn about the extensibility options of Power Pages and Unified Service Desk.

## Unified Service Desk extensibility

The **Unified Service Desk (USD)** is another extension technology for Dataverse that offers a framework for building contact center and call center solutions, as described in *Chapter 3, Understanding the Microsoft Power Platform Architecture*. The USD consists of the following main components:

- **USD packages:** The USD installs several solutions, along with a basic set of configuration data in the USD-specific tables. This configuration data directly influences the composition of the USD client.
- **USD client:** The USD client is a desktop application that is directly used by the contact center/call center agents.

The extensibility of USD consists of the following possibilities:

- **Configure** the content of the USD client (also called the **agent desktop**). The configuration makes it possible to specify integrated applications, events, **User Interface Integration (UII)** actions, action calls, standard workflows, and UI application automations. This type of configuration is performed by configuring the settings in the USD-related tables.
- **Integrate telephony** using the **UII Computer Telephony Integration (CTI)** framework. By doing this, it is possible to integrate an agent desktop solution with a number of CTI solutions from various telephony providers. The integration can contain inbound and outbound telephony automations. This customization type requires deep experience with CTI systems and custom development skills.
- **Develop** additional components for the USD, such as custom application adapters (DDA adapters), custom hosted controls, and custom panel types. All these customizations require custom development efforts.
- **Integrate** the USD with additional communication channels using the *Channel Integration Framework*.

The USD is an advanced technology that requires deep technical knowledge and programming skills.

### Important note



For further details about USD extensibility, please refer to the product documentation: <https://docs.microsoft.com/en-us/dynamics365/unified-service-desk/extend-unified-service-desk>.

With this, we have finished analyzing the extensibility possibilities for Dataverse-based solutions. In the next section, we will focus on the extensibility possibilities for canvas apps and Power Automate.

## Presenting Power Pages extensibility

Power Pages is an extension technology that provides publicly facing portal capabilities for Dataverse-based solutions. Provisioning, administering, and configuring Power Pages can be, to some extent, performed using administration portals and the Power Pages Studio. Using this approach, standard portals or Dynamics 365-related specific portals can be configured and managed.

The Power Pages technology consists of the following main components:

- **Microsoft Azure:** The runtime environment for Power Pages is implemented on a shared Microsoft Azure-based technology. The customer does not have access to the individual Azure services, nor do they need an individual Azure subscription to run the portals.
- **Dataverse configuration and metadata:** The content of the individual portal is fully configured within the Power Platform environment, in the Dataverse database. Configuration and customization can be performed by using the Power Pages Studio and by making direct configuration changes in the records of the Power Pages-related Dataverse tables.

For advanced extensibility, the Power Pages technology provides the following options:

- Use frontend extensibility by implementing changes in certain Power Pages-related Dataverse tables using HTML, CSS, JavaScript, and the **Liquid** open-source template language. Liquid can be used to customize portals in the web pages, content snippets, or web template tables.
- Use the **Portals Web API** to call the Dataverse Web API to trigger transactions within Dataverse. This capability makes it possible to extend the standard Portals behavior with custom Dataverse-related logic.

### Important note



You can learn more about the Liquid template language and the Portals Web API by referring to the product documentation at <https://docs.microsoft.com/en-us/powerapps/maker/portals/liquid/liquid-overview> and <https://docs.microsoft.com/en-us/powerapps/maker/portals/web-api-overview>.

Power Pages is a complex technology and there is a need for custom development skills when it comes to building tailored portal applications.

## Presenting Power Automate flows

Power Automate is dedicated to providing cross-application and cross-technology automation solutions. Compared to the previously described Dataverse automation capabilities, Power Automate is independent of Dataverse. Power Automate offers two main automation solutions:

- **Cloud flows:** The cloud flows technology is used to build automations across IT systems and solutions, having an API, which can be connected using some of the *Power Platform data connectors*.
- **Desktop flows:** The desktop flows technology is used to automate IT systems and solutions that have no possibility to provide an API, so Power Platform connectors cannot be used. Desktop flows are automated by replicating the processes a human user would perform on the IT systems.

Both automation technologies are solution aware so that the flows can be distributed between environments within Power Platform solutions.

Let us now provide some more details about the two types of Power Automate technologies.

### Cloud flows

Compared to Dataverse workflows, the capabilities of Power Automate cloud flows are broader, so more types of automations can be implemented. For example, the cloud flows can perform record deletions, as well as retrieve individual records or groups of records from Dataverse and process them in a loop.

In order to build a Dataverse automation with cloud flows, the respective Dataverse data connector needs to be used.

There are currently two different types of Dataverse connectors:

- **Dataverse:** This connector can only be used from within a Power Platform solution. For this connector, there is no need to specify the Power Platform environment; the flow automatically connects to the current environment. After deploying within a Power Platform solution to a different environment, the connector updates automatically.

- **Dataverse (legacy):** This connector can be used for Power Automate flows that are created within a Dataverse solution, or outside of a Dataverse solution. The trigger, as well as any action step in the flow, can be configured for a selected Power Platform environment, or for the current environment. When deployed to a different Power Platform environment, the connection to the new environment must be eventually manually updated. This connector is considered deprecated and should not be used for new solutions.

Both Dataverse connectors can be used as **triggers** for standard Dataverse events such as *record creation*, *record update*, or *record deletion*. There are some additional interesting triggers that can be used in the automation scenarios, like triggering a flow from a BPF or from a *custom action*, or on-demand for a *selected record(s)* from model-driven apps.

The connectors provide a selection of **actions**, which can be used to implement certain automation procedures:

- **Create a Dataverse record:** This action creates a record in a Dataverse table.
- **Update a Dataverse record:** This action updates a record in a Dataverse table.
- **Delete a Dataverse record:** This action deletes a record in a Dataverse table.
- **Get a Dataverse record:** This action makes it possible to retrieve a single record and use its values for the business logic.
- **List Dataverse records:** This action makes it possible to retrieve a list of records and use its values for the business logic in a loop.

The following are the additional actions, provided only in the modern Dataverse connector:

- **Execute a changeset request:** This action can perform multiple Dataverse create, update, or delete actions in the context of a transaction so that if any of the actions fail, the whole changeset will be rolled back.
- **Get or upload a file or image:** This action can retrieve or upload a file or image from/to a Dataverse column.
- **Perform a bound or unbound action:** This action can call a Dataverse custom action and use the returned results in the business logic.
- **Predict:** This action can call an existing AI Builder model and use the returned results in the business logic.
- **Relate or unrelated records:** This action can create or remove a link between two table records with a 1:N relationship.

**Important note**

You might still see the Dataverse automation **Dialog** type available when creating a new process. However, Dialogs are deprecated and should not be used for new Dataverse implementations anymore.

## Desktop flows

As explained in the introduction, desktop flows are used where cloud flows cannot be used due to the lack of an existing API. The **Power Automate Desktop** technology is used to fill this gap using something called **robotic process automation (RPA)**. The idea is that automation can be also achieved by simply replicating a series of steps a human user would perform to conclude certain business processes. So, the solution is to create this series of steps and then let this series be replayed automatically on a piece of infrastructure, usually a physical or virtual computer.

The main automation idea behind Power Automate is to primarily use cloud flows whenever possible and resort to desktop flows only when cloud flows cannot be used. Cloud flows, however, should always be the leading technology, and desktop flows should be typically started from a “master” cloud flow, as illustrated in the following diagram:

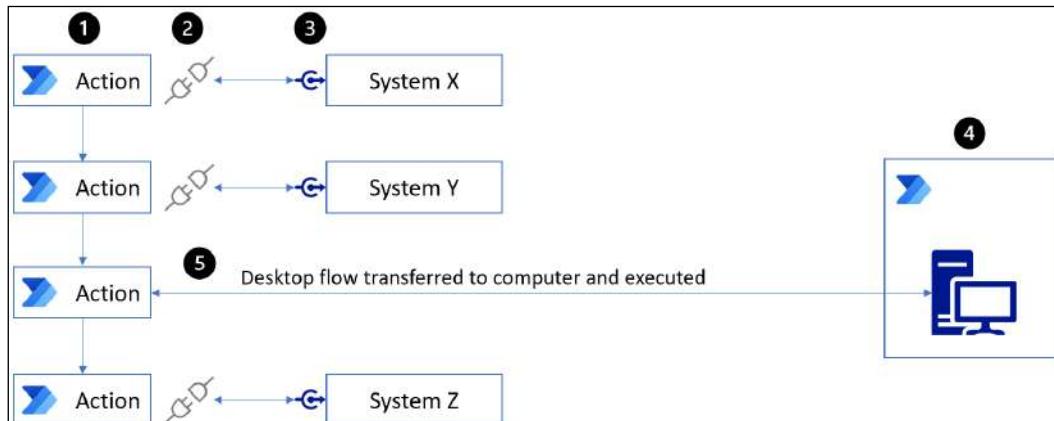


Figure 8.16: Automation example with desktop flow

The above diagram describes the overall functionality of Power Automate Desktop in the following way:

- A Power Automate cloud flow (1) implements a cross-application and cross-technology automation process.

- The cloud flow is using the Power Platform data connectors (2) to connect to IT systems and solutions having an API (3).
- There is a legacy system in operation (4), which cannot be connected to via API.
- The cloud flow performs an upload of a previously configured desktop flow (5) from the cloud to a dedicated computer, where the Power Automate Desktop product is installed. After that, the desktop flow is started, equipped with input parameters coming from the cloud flow. The desktop flow is executed and eventually returns some values as output parameters. The cloud flow can use these return values in the next steps of the automation procedure.

The Power Automate Desktop technology, as the name already suggests, is not a cloud-only service, but contains some locally installable components, namely:

- The **Power Automate Desktop** itself, which is a configuration tool and a runtime at the same time.
- The **browser extensions** for the major browsers Edge, Chrome, and Firefox. The extensions are required for desktop automation of web-based solutions.
- The **machine runtime app**, which is required to handle the hybrid connection from the Power Automate cloud to the respective local computer. This is the replacement for On-Premises Data Gateway, which was used for this purpose in the previous versions of Power Automate Desktop.

The desktop flows can be created in one of the following ways, but they can be also combined:

- **Manual creation:** The Power Automate Desktop contains a large number of different desktop actions, which can be manually configured to represent the steps of the business process.
- **Recording:** The series of steps can also be recorded, while a human user is performing the steps on the specified IT system. For example, the flow can perform any mouse movements, clicks, and keystrokes inputted by the user.

Either way, the desktop flow can be polished to remove or correct wrong steps, then equipped with input and output variables and prepared for the intended purpose.



### Important note

The use of Power Automate Desktop within the automation solution requires additional licensing since the product is not included in the usual Power Apps or Power Automate licenses. For further details, please refer to the Power Automate licensing documentation: <https://learn.microsoft.com/en-us/power-platform/admin/power-automate-licensing/types>.

Lastly, it is important to understand that desktop flows can run in two different modes:

- **Attended mode:** In this mode, a human user signs in to a computer and works with the desktop flows, for example, to develop and test them.
- **Unattended mode:** This is the production mode, where the computer is not attended by any human user – it signs in and executes the desktop flows autonomously. It is important to design the proper infrastructure setup for the expected volume of desktop flow executions. The machine runtime app supports a single-computer as well as a multi-computer setup for this.



### Important note

For further details about Power Automate Desktop, please refer to the product documentation: <https://learn.microsoft.com/en-us/power-automate/desktop-flows/introduction>



### Tip

For a detailed introduction to how to create flows in Power Automate, consider checking out *Workflow Automation with Microsoft Power Automate, Second Edition*, by Aaron Guilmette.

## Process advisor

In the context of Power Automate, it is worth mentioning a new service called **process advisor**. The purpose of the service is to track the activities of business users using various IT systems and analyze them to provide guidance on how to improve and optimize them using automation. The service is based on the Power Automate Desktop technology.

This was the last section dealing with low-code/no-code extensibility within Power Platform solutions.

In the following sections, you will learn how to extend Dataverse with code on the client side, as well as on the server side. Let us start first with the client-side extensibility.

## Presenting canvas apps and Power Automate extensibility

Canvas apps and Power Automate are Power Platform components that don't provide any ready-to-use solutions. Both of these solution components are platforms for building mobile applications or automation and integration scenarios.

In the following sections, you'll understand how to use these technologies beyond the usual app or flow design.

### Canvas apps and Power Automate customization

**Canvas apps** are typical representations of the *low-code/no-code* approach to building apps primarily for mobile devices, though canvas apps can also run on the usual desktop devices. Canvas apps have the following main characteristics:

- The primary use of canvas apps is to implement mobile applications, thus leveraging and connecting to existing IT systems and deploying within the organization for authenticated users only.
- Canvas apps are built using a typical mobile app development approach. This is done by creating a certain number of screens and placing the required UI and functionality on these screens.
- Navigation between screens and all other business logic is implemented using an Excel-like expression language called **Power Fx**. The use of this simple expression language should open the development of canvas apps also to non-IT-professionals.
- Connections to any data sources and IT systems are implemented with the use of data connectors. There are a high number of public connectors on the market already, though if needed, custom connectors can be developed and used.

Canvas apps can be further customized using the following solution approaches:

- **Use Power Apps Component Framework controls:** PCF controls, as described in the *Understanding Dataverse server-side extensibility* section about Dataverse extensibility, can be used to extend the user interface of canvas apps as well.

- **Use canvas app components:** Canvas apps make it possible to build components directly in the canvas apps development environment (Power Apps Studio). These types of custom components can be made part of a component library and reused within canvas apps. Canvas app components can only be used in canvas apps, not in model-driven apps like the PCF components.
- **Build custom connectors:** There is a large number of public connectors for canvas apps, Power Automate, and Azure Logic Apps already. However, if no public connector is available for the technology that needs to be integrated, you can develop custom connectors.

You learned about **Power Automate** in the previous section of this chapter. Power Automate can also be customized using custom connectors. There is no other extensibility of Power Automate using code, besides the use of custom connectors.

## Power Fx

**Power Fx** has been briefly mentioned previously in this book, but now is the right time to talk more about this new and important Power Platform technology. The main idea behind Power Fx is to provide an easy-to-use Excel-like expression language, which should sound familiar to every business user and so support the low-code nature of the Power Platform.

Power Fx started as a nameless Excel-like expression language, supporting the application functionality in canvas apps. As of now, the language has the name “*Power Fx*,” but it has also started to support more than just canvas apps. The areas where Power Fx can be used are the following:

- **Canvas apps functionality:** This is the basic use of Power Fx. Within canvas apps, this use is necessary for every business functionality.
- **Command bar functionality:** As described earlier in this chapter, there is a new way to configure the functionality behind command bar controls. Currently, it is possible to create new command bar controls within Power Apps Maker Portal and equip them either with JavaScript or with Power Fx commands to implement their functionality. As of writing this book, this capability is still in preview.
- **Calculated columns in Dataverse:** As we learned earlier, there are three main types of Dataverse columns: **simple** columns, which are used in the usual way, and then **calculated** and **rollup** columns, which are both calculated automatically through the Dataverse platform. The formulas for these calculations currently use the proprietary configuration capability. As of writing this book, there is a new possibility to configure the formulas using Power Fx, but again, it is currently in preview.

The Power Fx language makes it possible to use either a *declarative* or *imperative* programming style. The number of expressions is already quite high, and the language is evolving with time.



**Important note**

To learn more about Power Fx, please refer to the product documentation: <https://learn.microsoft.com/en-us/power-platform/power-fx/overview>.

In the following section, we will cover some more details about creating custom connectors for canvas apps or Power Automate flows.

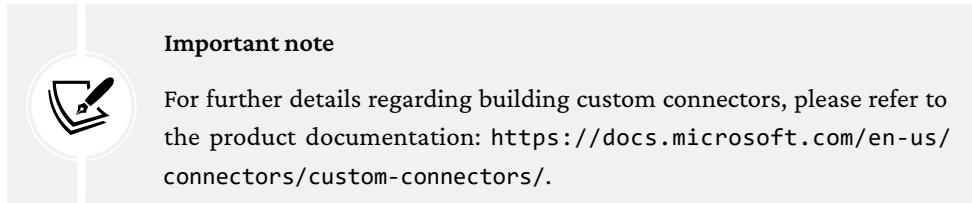
## Building custom connectors

Custom connectors are wrappers around an existing REST API that expose certain interface capabilities of IT systems. Custom connectors can be used to provide connection to the cloud as well as on-premises IT systems. For on-premises IT systems, the **On-Premises Data Gateway** needs to be implemented.

The process of developing and configuring a custom connector consists of the following steps:

1. **Develop the API:** Any custom connector is based on an existing physical interface. If there is no compatible API available yet, it must be developed first. In the world of Microsoft cloud solutions, the best technologies to develop the necessary physical API are Azure Functions, Azure Web Apps, and Azure API Apps.
2. **Configure API security:** In order to be configured as custom connectors, the API must support one of the standard authentication methods: OAuth 2.0, basic authentication, or an API key.
3. **Configure the custom connector:** Once the API has been developed, the custom connector needs to be configured based on the API. The custom connector is configured in the Power Apps Maker Portal. You can configure the connector by creating it from scratch, importing an OpenAPI definition, a Postman collection, from GitHub, or creating it directly from an existing Azure service.
4. **Use the custom connector:** Once the custom connector has been configured within the Power Platform environment, it can be used for canvas apps and Power Automate flows.

5. **Share the connector:** Custom connectors are not automatically available to all licensed users in the tenant, like the public connectors. The custom connector needs to be shared with other makers for them to use them.



If the author of a custom connector is interested in publishing the connector for all users of the Microsoft cloud services, they can run through a certification process. After successful certification, the connector can be deployed to the public repository of connectors.

## Presenting Power BI extensibility

You learned a lot about the purpose of Power BI, its architecture, ALM, and security in the previous chapters of this book. Power BI offers a broad range of standard capabilities for building interactive reports and dashboards.

In case you need to extend the standard product's capabilities, Power BI offers the following extensibility options:

- **Build custom visuals:** Power BI offers a number of default visuals, and another broad selection of visuals is available from AppSource. If a specific visual is not available, there is a possibility for custom development.
- **Manage Power BI with code:** The Power BI API offers certain management possibilities, such as embedding Power BI content into various types of applications or pushing data into Power BI datasets.
- **Build custom Power BI data connectors:** Power BI offers the possibility to build custom connectors to connect to data sources not available in the standard Power BI service. A large number of established data sources are available for use in Power BI. However, there might be situations where a data connector for a specific technology is required but not publicly available. Power BI custom connectors are **completely different** from the data connectors for canvas apps and Power Automate.



### Important note

For further details regarding Power BI extensibility, please refer to the product documentation: <https://docs.microsoft.com/en-us/power-bi/developer/>.

Resources for building Power BI data connectors can be found in the following GitHub repository: <https://github.com/Microsoft/DataConnectors>.

Now that we've covered all the theoretical knowledge about the extensibility options available for Power Platform components, let's focus on the proven best practices so that we can use these options in the best way and avoid the usual pitfalls.

## Power Platform extensibility best practices

In the following sections, you will learn about some of the best practices for client-side as well as server-side extensibility for Dataverse-based solutions.

### Dataverse client-side interface extensibility

In this chapter, you have learned a lot about various client-side Dataverse extensibility and automation options. In this section, we will provide several best practices for using the various extensibility options available to you, specifically from a performance point of view.

It is always important to keep the performance of the Dataverse-based solution in mind. Client-side extensibility options can have a significant impact on the final solution's performance. Try to implement the following best practices when designing the extensibility to avoid the most common sources of poor performance:

- Design the Dataverse table forms by providing only the necessary controls; avoid overwhelming the forms with a large number of complex controls. Very large forms usually load slower, which obviously leads to poor acceptance.
- Use more tabs with a smaller number of sections and controls, rather than a large number of sections and controls on a single tab. Using this approach can help the end user start working with the data on the form faster, without the need to wait for all the content on the tab to load.
- When using embedded iFrame form components, consider loading the content asynchronously to speed up the form's load time. This can be done using some advanced JavaScript code. An easier way to handle iFrames is just not to place them on the first form's tab so that the content can load in the background, while the end user is working on the most important data.

- Carefully consider using embedded canvas apps on Dataverse forms for performance reasons – do not place canvas apps on the first form tab.
- Use PCF controls rather than web resources, since they load fast in the main form load event.
- Use Dataverse business rules instead of JavaScript event handlers whenever possible. Business rules are an integral part of the Dataverse platform and usually perform faster than JavaScript event handlers.
- Consider the most performance-sensitive client-side events, such as OnLoad, and avoid registering complex JavaScript event handlers for those events. Complex and long-running JavaScript event handlers can significantly slow down the form's load performance.
- For any communication with the Dataverse API, use the client-side Web API instead of SOAP-based calls. It is faster and much easier to implement.
- Avoid using multiple calls to several external web service endpoints in the JavaScript event handlers, especially for performance-sensitive client-side events. Having multiple calls in an event can significantly slow down performance since every call adds up additional time for sending the request, server-side processing, and receiving the response. For high-latency scenarios, this can have an extremely negative performance impact. Consider developing a specific wrapper web service to consolidate the whole server-side business logic into a single service that can be called from JavaScript instead.

In the next section, we will cover the server-side extensibility best practices.

## **Dataverse server-side extensibility**

There are many possibilities for the server-side extensibility of Dataverse. In the following sections, you will learn how to select the most suitable Dataverse APIs, how to select the best extensibility and automation options, and what should be considered when building a performance-optimized Dataverse solution.

## **Dataverse API selection**

When planning to build Dataverse server-side extensibility, there are three possibilities regarding the use of APIs and tooling:

- **Plug-ins and custom workflow actions:** For these types of extensibilities, the only option is to use the organization service, which is part of the execution context. The benefit of doing this is having the respective DLL assemblies available, which makes the development process and the existing pre-authentication of any communication easier.

- **External .NET-based applications:** For these types of extensibilities, the most suitable approach is to use the **XRM Tooling** assemblies, since they provide a set of DLL assemblies with a lot of ready-made capabilities for custom development. For example, the whole authentication complexity is covered by one of the classes contained in XRM Tooling.

It is recommended to use OAuth authentication for increased security instead of Office 365 authentication, even though Office 365 authentication is available within XRM Tooling.

Another option, when not using the XRM Tooling assemblies, is to resort to the modern Web API, OAuth authentication, and either the standard .NET HTTP client or any of the third-party supporting libraries.

- **External application, not .NET based:** For these types of extensibilities, the only option is to use the Web API, since Microsoft does not provide any support for SOAP from non-.NET solutions.

Always select the proper authentication method against the Dataverse API. The use of legacy Office 365 authentication is possible for SOAP-based endpoints; however, it is considered less secure. Use OAuth whenever possible, even for SOAP.

#### Important note



There are good community sources available that provide supporting information for various frameworks and programming languages for using the Web API against the Dataverse API. Please refer to the *Further reading* section at the end of this chapter for more information.

In the next section, we will present a comparison between the various approaches for using extensibility and implementing automations.

## Extensibility and automation options

In this chapter, you have learned a lot about various server-side Dataverse extensibility and automation options. In this section, we've provided a summary and overview of the various options available, highlighting the typical usage scenarios and technology constraints:

- **Dataverse business rules:** Simple automations based on a single table record. They can replace JavaScript client scripting to some extent, and they also work on the server side.
- **Dataverse workflows:** Medium-complexity automations that can only run on the server side and can execute synchronously or asynchronously. For asynchronous scenarios, these can be replaced with Power Automate.

- **Dataverse custom actions/custom APIs:** Medium-complexity automations that can only run on the server side. They do not have their own triggers, so they need to be triggered from a Dataverse workflow, business process flow, or code.
- **Dataverse business process flows:** Simple user interface-based automations. They provide guidance to end users in terms of following standard business processes. They can include various embedded automations using Dataverse workflows, Dataverse custom actions, or Power Automate flows.
- **Power Automate:** Medium-complexity automations that can only run on the server side and only asynchronously. It has a broader range of capabilities compared to Dataverse workflows. It can run as an event handler but can also be triggered manually from the Dataverse application's command bar or from Dataverse business process flows.
- **Plug-ins:** High-complexity automations that can only run on the server side and only as event handlers; they cannot be triggered manually. They can be implemented for synchronous or asynchronous execution and need to be developed using custom code in .NET. Plug-ins should be considered when any of the low-code/no-code approaches are unable to cover complex business requirements.
- **Custom workflow actions:** High-complexity automations that can only run on the server side and must be triggered from Dataverse workflows. They need to be developed using custom code in .NET. The usage scenario is the same as it is for plug-ins. The difference is that using custom workflow actions provide the flexibility needed to change the base business logic without code within the Dataverse workflow, while for plug-ins, every change in the business logic must be implemented by modifying the code.
- **Azure Service Bus integration:** Can be used to implement remote automation or outbound hybrid integration scenarios in asynchronous mode only. Requires a separate Microsoft Azure license. The listener component needs to be developed using custom code in .NET.
- **Webhook integration:** Can be used to implement remote automation or outbound integration scenarios in synchronous or asynchronous mode. The functionality can be implemented in the cloud, but also on-premises.
- **Building external applications:** Can be used to implement a variety of business requirements from batch jobs, through enterprise integration scenarios, to every kind of alternate client application solutions.

In the next section, you will learn about some of the best practices in terms of performance optimization for Dataverse applications.

## Performance impact

When architecting and designing server-side extensibility, it is very important to keep the performance of the future solution in mind the same way as you would for client-side extensibility. There are certain solution approaches that should be limited or entirely avoided since they have a known negative performance impact:

- **Synchronous Dataverse workflows** should only be used when the requirement dictates to have the result of the workflow being processed available immediately on the user interface. It is recommended to reduce the complexity and duration of synchronous workflows to a minimum.
- The same recommendation applies to **synchronous plug-ins**.
- Implementing business logic using plug-ins registered for the `Retrieve` or `RetrieveMultiple` events should be considered very carefully. Such plug-ins are known to have a serious negative performance impact. If required, performance optimization of the plug-in's code is necessary.

With this section, we have finished looking at Power Platform's extensibility best practices. Next, we will look at our fictitious Power Platform customer and see what they have learned and decided to do for their project's implementation.

## Contoso Inc. Power Platform solution design

Contoso Inc. has made itself familiar with all the extensibility options for the Power Platform cloud services and components. Together with their implementation partner, Proseware Inc., they are working on the solution design, for which they have made a series of decisions.

In this section, we will describe their design decisions in more detail.

### Model-driven apps

The project teams at Contoso Inc. have understood the concept of model-driven applications. They have decided to use this capability to design custom, user-group-specific model-driven apps based on the Dynamics 365 capabilities they are planning to use. This will make the adoption of the Dynamics 365 workloads much easier since every user group will have just their main capabilities available in the tailored model-driven apps. Contoso Inc. will use the assignment of security roles to distribute the right apps to the right user groups.

Contoso Inc. has made a general decision not to use the SQL Server Reporting Services technology for reporting purposes within model-driven apps because they believe this to be a legacy technology with no clear future. Instead, they've decided to use views, charts, and dashboards for simple reporting and Power BI for advanced reporting and analytics purposes.

## Automations

Contoso Inc. has analyzed the various automation possibilities for Dataverse and Dynamics 365 applications and decided to use *low-code/no-code* automations in every situation, where the capabilities of these automations will permit. The following automation types will be used:

- Dataverse business rules
- Dataverse business process flows
- Power Automate

Contoso Inc. will try to completely avoid using synchronous Dataverse workflows because they have understood their potential negative impact on performance. Since the Power Platform solution will be used globally from a single Microsoft data center, the effect on the latency time in certain regions would be noticeable.

Instead, they will use Power Automate for all medium complexity automations. For possible high-complexity automations, Contoso Inc. will use asynchronous workflows or Power Automate flows with custom workflow actions to keep the highest level of flexibility for possible future changes in the business logic. It will be recommended to avoid the use of plug-ins, whenever possible. Contoso Inc. will follow all the performance-related best practices for server-side extensibility.

## Client-side extensibility

Contoso Inc. has also made itself familiar with the client-side extensibility options and agreed on certain rules for their project.

They decided to use the following client-side extensibility options:

- If required, they will use standard custom controls instead of default controls.
- If there is no suitable standard custom control, the development of PCF controls will be considered.
- JavaScript event handlers will be used only if the required business capability cannot be implemented with Dataverse business rules. Every JavaScript event handler must be carefully optimized for performance.

- If required, the use of command bar extensibility will be considered a viable option.

Contoso Inc. decided not to use web resources and canvas apps for extending the Dataverse user interface. They learned that using HTML web resources is not recommended for building UI extensions anymore due to the comparative advantages of using other controls such as standard custom controls and PCF. As for canvas apps, the implementation approach for embedding them into model-driven apps is somewhat complicated. It would be better to look into this approach later as it evolves.

## **Server-side extensibility and integrations**

Contoso Inc. has understood the capabilities of the Dataverse API, as well as the various standard outbound integration options, and decided to use Power Automate and Azure Functions for all possible batch processing requirements.

For now, the integration architecture and design will be postponed until the integration capabilities, patterns, and requirements have been analyzed in more detail. Currently, there are no plans to build alternate clients or integrate with third-party portals.

## **Other design decisions**

Contoso Inc. will consider using Power Pages for implementing certain publicly facing capabilities using the default possibilities of this technology. The use of canvas apps is under investigation; there might be good use cases to implement certain single-purpose internal applications using this technology. Power BI will be used as the primary analytics and reporting solution for the whole Power Platform implementation. There are currently no plans to use code-based extensibility for Power BI.

Contoso Inc. is satisfied with the outcome of this project phase. They now have full clarity regarding how to implement the various components of their Power Platform solution and are now ready to start investigating, planning, architecting, and designing the integration of their solution.

## **Summary**

In this chapter, you have learned about designing, building, and extending Dataverse applications, implementing standard and custom automations, and using code to provide high-complexity extensibility requirements. You have seen how to extend Dataverse applications on both the client side and the server side. Furthermore, you have learned about the various extensibility options that are available for canvas apps, Power Automate, and Power BI.

With this knowledge, you should be able to prepare a robust design for your Power Platform solution, while also following all proven best practices.

In the next chapter, we will focus on all the topics related to integrating Power Platform solutions with other IT systems and solutions.

## Further reading

With the rise of the Dataverse Web API, as one of the later extensions of the Dataverse API, the possibilities of developing applications integrated with Dataverse are now much better for non-Microsoft developers. The Dataverse Web API is an industry standard that can be used from within a lot of popular programming languages. There is no official support from Microsoft, but rather a very dynamic community that provides useful support and resources.

For some of the most common languages and their resources, go to the following links:

- Various OData libraries: <https://www.odata.org/libraries/>
- Java: [http://rest-examples.chilkat.io/dynamics\\_crm/java/default.cshtml](http://rest-examples.chilkat.io/dynamics_crm/java/default.cshtml)
- Perl: [http://rest-examples.chilkat.io/dynamics\\_crm/perl/default.cshtml](http://rest-examples.chilkat.io/dynamics_crm/perl/default.cshtml)
- PHP: [http://rest-examples.chilkat.io/dynamics\\_crm/phpExt/default.cshtml](http://rest-examples.chilkat.io/dynamics_crm/phpExt/default.cshtml)
- Python: [http://rest-examples.chilkat.io/dynamics\\_crm/python/default.cshtml](http://rest-examples.chilkat.io/dynamics_crm/python/default.cshtml)
- Node.js: [http://rest-examples.chilkat.io/dynamics\\_crm/nodejs/default.cshtml](http://rest-examples.chilkat.io/dynamics_crm/nodejs/default.cshtml)

It is really important to fully understand what the supported and unsupported extensibility options for Dataverse solutions are. You can find full recommendations regarding this important topic in the following Microsoft product documentation article: <https://learn.microsoft.com/en-us/power-apps/developer/data-platform/supported-customizations>.



# 9

## Microsoft Power Platform Integration

One of the most complex parts of any large enterprise Power Platform implementation is usually integrating the solution into the existing IT landscape of the organization. While security integration was covered extensively in *Chapter 7, Microsoft Power Platform Security*, this chapter is dedicated to integrating the Power Platform solution with Microsoft 365, Microsoft Azure, and with customers' existing IT systems and solutions. It is important to understand which integrations can be achieved easily with configurations and which integrations need to use additional components and/or custom development to be able to create a proper integration design. As always, the *low-code/no-code* approach is the best option, but for very complex scenarios, the use of custom development methods needs to be considered.

In this chapter, we are going to cover the following main topics:

- Power Platform integration overview
- Integration with Microsoft 365 and Microsoft Azure
- Frontend integration patterns and solution approaches
- Backend integration patterns and solution approaches
- Other Power Platform integrations
- Power Platform integration best practices
- Contoso Inc. Power Platform integration design

## **Contoso Inc. designing the Power Platform integration**

Contoso Inc. is progressing very well with the design and development of the Power Platform solution components. They were able to gain the necessary clarity on the extensibility options and have taken a number of architecture and design decisions in terms of how to build the Power Platform components. The next step in their journey is to understand how a Power Platform solution can be integrated into their existing IT landscape.

Contoso Inc. is a large global company with a lot of existing IT systems, many of which will still be used after the Power Platform solution is implemented. Most of these IT systems are operated on-premises in their own data centers, but Contoso Inc. is also using some specialized cloud services. They have assessed their cloud maturity as medium, having some experience with operating cloud solutions, but the planned Power Platform solution will be the largest cloud project so far. They are eager to understand the integration possibilities of Power Platform and whether they are aligned with their cloud integration strategy.

## **Getting an overview of Power Platform integration**

There are no isolated island Power Platform implementations. Every implementation, especially large and complex ones for an enterprise customer, needs to be integrated into the existing IT landscape of the organization. The solution can also benefit from the standard integrations with various Microsoft cloud services. There are many reasons why a Power Platform solution should be integrated, so let's mention some of them:

- A Power Platform solution is often about communicating and interacting with customers, vendors, patients, citizens, and others. In these cases, it is important to integrate some electronic communication channels to support the solution.
- A Power Platform solution can be a data master for certain data domains, but it certainly needs to accept a lot of data that's mastered in other systems. A consolidated solution should always avoid entering the same data into several IT systems manually by providing mutual data integration.
- A Power Platform solution should be able to provide additional business value by easily integrating with other existing Microsoft cloud services to achieve a synergy effect with low effort.
- It can be required for a Power Platform solution to provide integrated document management.

- A Power Platform solution should allow us to contribute to an overall data warehouse solution with relevant data.
- A Power Platform solution should be able to offload historical data into an archiving solution for operational or legal reasons.

These are just some of the possible reasons that highlight the need to integrate a Power Platform solution with other existing or new IT systems and solutions.

In the following sections, you will learn about many standard integrations that are available within Power Platform just by enabling and configuring them, as well as the possibility of using custom integrations with various systems and technologies.

Let's start with the integrations available out of the box or through using simple configurations.

## Integrating with Microsoft 365 and Microsoft Azure

Since Power Platform is a very important member of the family of Microsoft cloud services, it is a legitimate expectation that a Power Platform solution can easily integrate with many of them. In this section, you will learn about the integration possibilities of Power Platform with the two other services in the Microsoft cloud: Microsoft 365 and Microsoft Azure.

We'll start by explaining how certain Dynamics 365 applications are implicitly integrated with other Microsoft cloud services.

### Introducing implicit Dynamics 365 integrations

As you learned in *Chapter 1, Microsoft Power Platform and Microsoft Dynamics 365 Overview*, there is a big family of Dynamics 365 apps from Microsoft. These applications cover certain business workloads such as sales management, marketing management, customer service management, and so on. Many, if not all, of those applications provide certain capabilities by using *implicit integration* with Microsoft Azure or Microsoft 365.

In most cases, these capabilities are fully transparent to the end user, and there is also no need for the customer to have a separate subscription for Microsoft Azure to be able to use them. However, there are certain Dynamics 365 applications, or add-ons, where a separate subscription is necessary.

The following is an overview of some of the implicit integrations of the various Dynamics 365 and Dataverse applications, where *no additional license is necessary*:

- **Dynamics 365 Sales:** In this app, the whole *Sales Insights* capability is implemented using integration with Microsoft Exchange and Microsoft Azure services.

- **Dynamics 365 Marketing:** In this app, the *marketing forms* and *pages* are hosted in a portal solution, and the *mass mailing* capability along with the *built-in analytics* are implemented using Microsoft Azure services.
- **Power Pages:** This solution consists of the metadata part, hosted within a Dataverse environment and a *web application*, running in a shared Microsoft Azure environment.

A customer can use these capabilities without the need to know about the underlying cloud infrastructure. They neither need a specific Microsoft Azure license, nor any access to the shared infrastructure hosting the services.

Another approach is an implicit integration where *the customer must provide their own additional cloud subscription* in order to use the capability. The best example of this is the **Dynamics 365 Connected Field Service**. This application is an add-on for Dynamics 365 Field Service, extending this application with the integration of IoT devices. To use this add-on, the customer must either use their own Microsoft Azure license to provision **Azure IoT Hub**, including all its additional services, or subscribe to use **Azure IoT Central**, which is a SaaS type of cloud service. In this case, the customer has access to the additional cloud environment, and they are responsible for provisioning and maintaining the environment.

In the next section, we will discuss the explicit integration between Dataverse or Dynamics 365 and various Microsoft 365 services.

## Integrations with Microsoft 365 services

Dynamics 365 and Dataverse applications can benefit from a lot of out-of-the-box integrations with Microsoft 365 services. In most cases, this integration can be achieved using a very simple configuration setting. In the following sections, you will learn more about these standard integrations.

### Integrating with Exchange

Dataverse can be integrated with both **Exchange Online** and **Exchange On-Premises**. For the Exchange Online integration, both services must reside in the same tenant. A limited integration (emails only) can be established with non-Exchange mail servers of the SMTP/POP3 type. Integration with Exchange makes it possible to synchronize *emails*, *appointments*, *tasks*, and *contacts* between those two systems. The integration approach is shown in the following diagram:

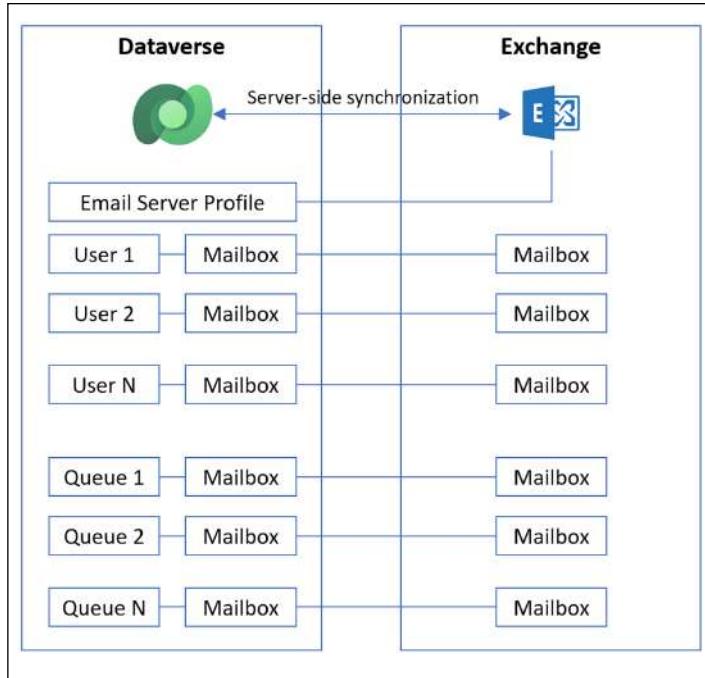


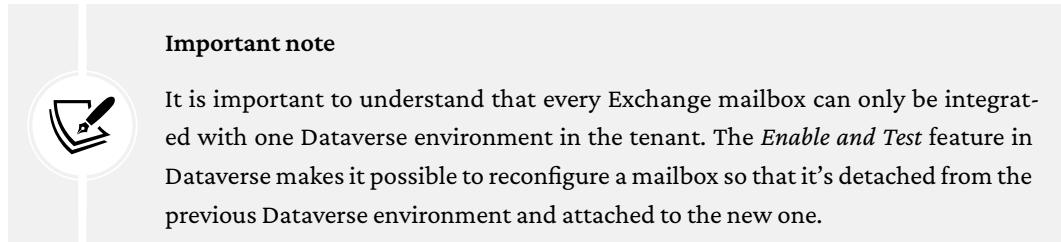
Figure 9.1: Dataverse integration with Exchange

As we can see, on the Dataverse side, there is a table called **Email Server Profile** that specifies the connection to an **Exchange** (or other) server. It is possible to have a heterogeneous configuration with multiple email server profiles of different types (for example, an Exchange On-Premises server for the majority of the existing users, together with an Exchange Online profile, for some of the users, already migrated to the cloud).

Every user or queue automatically has a corresponding **Mailbox** record created and assigned, which represents the connection between the Dataverse user and the corresponding email account in **Exchange**. The process of establishing this integration is performed in the Power Platform admin center and consists of the following steps:

1. Configure the default **Exchange integration settings**.
2. For Exchange Online integration, the **Email Server Profile** is automatically created. For Exchange On-Premises or SMTP/POP3 integrations, an **Email Server Profile** needs to be configured with all the required settings.
3. Every individual **Mailbox** record needs to be configured. The configuration consists of *approving* the mailbox and then *enabling and testing* it.

Once you've completed these configuration steps, the integration will be established and the Dataverse application users can benefit from synchronizing important data such as emails, appointments, contacts, or tasks between those two systems.



Now, let's have a look in the following section at the integration with **SharePoint**.

## Integrating with SharePoint

Power Platform applications can be integrated with **SharePoint** to achieve integrated document management for collaborative work on relevant documents. Since the integration between those two systems has a very important security aspect, you can find more details by reading *Chapter 7, Microsoft Power Platform Security*. As for Exchange Online, SharePoint Online must be in the same tenant as the Power Platform environment. A hybrid integration with SharePoint on-premises is also available.

The integration approach is shown in the following diagram:

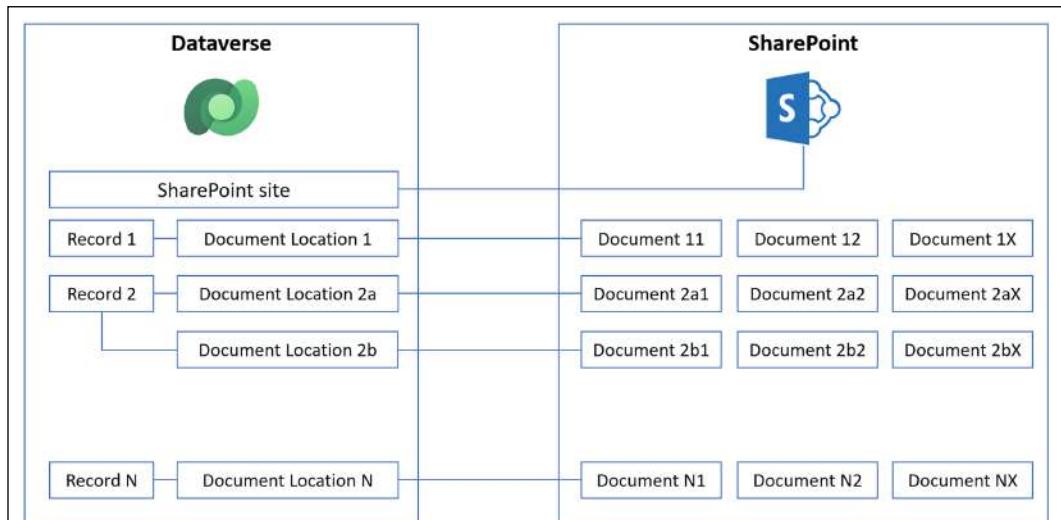


Figure 9.2: Dataverse integration with SharePoint

As you can see in the diagram, it is possible to establish an integration to a selected SharePoint site, the URL of which is stored in the **SharePoint site** record. For every Dataverse table, for which the integration was enabled, there can be one or multiple document locations, represented with the records in the **Document Location** table. The document location represents a folder on the SharePoint site, in which there can be multiple documents saved. Having multiple document locations per Dataverse record can be either performed by the end user, or by establishing additional integrations, for example, with OneDrive or Teams, as described in the next sections of this chapter.

The process of establishing integration is performed in the Power Platform admin center and consists of the following steps:

1. Create a dedicated SharePoint site. This site will be used for storing and managing various documents from within the model-driven apps.
2. Perform the configuration in Dataverse using the dedicated SharePoint site. Part of the configuration is to decide which Dataverse tables will be enabled for the integration.

After this simple configuration, integrated document management will be available from the main table forms for records of each enabled table.

Now, let's have a look at another integrated document management feature, this time for storing private documents.

## Integrating with OneDrive

Another solution that provides integrated document management for Power Platform applications is integration with **OneDrive**. The biggest difference between SharePoint and OneDrive is that while SharePoint is a collaborative platform where normally every user has access, OneDrive is a private document storage software where every user has their own protected storage area.

In order to use OneDrive integration, SharePoint integration must be established first. OneDrive integration can be easily configured in a few simple steps within the Power Platform admin center:

- Establish the SharePoint integration as described in the previous section, if not done already.
- In the **Document Management** area in the administration, click on *Enable OneDrive*.

For the end user, both the SharePoint and the OneDrive integrations are presented in a unified user experience. The user can see all the attached shared documents in SharePoint, along with all the attached private documents in OneDrive in one single list, as shown in the following screenshot:

The screenshot shows a Microsoft Dynamics 365 Opportunity record for '4G Enabled Tablets'. The top navigation bar includes 'Opportunity Sales Process' (Active for 3 years), 'Qualify', and 'Develop'. Below the navigation is a ribbon with tabs: Summary, Product Line Items, Quotes, General, Field Service, **Files** (underlined), LinkedIn Sales Navigator, and Related. A 'Document Associated Grid' is displayed, showing three files: 'Confidential Opportunity Backgroud.pptx' (OneDrive), 'Opportunity Calculation.xlsx' (SharePoint), and 'Opportunity Summary.docx' (SharePoint). The grid has columns for Name, Modified, Modified by, and Source.

Name	Modified	Modified by	Source
Confidential Opportunity Backgroud.pptx	31.07.2020 15:33	Robert Rybaric	OneDrive
Opportunity Calculation.xlsx	31.07.2020 15:32	Robert Rybaric	SharePoint
Opportunity Summary.docx	31.07.2020 15:32	Robert Rybaric	SharePoint

Figure 9.3: Integrated SharePoint and OneDrive document management

As we can see, there is a list of associated documents attached to an opportunity. The list contains shared documents stored in SharePoint, as well as confidential documents stored on OneDrive.

## Integrating with Microsoft Teams

Another important integration provided for Dynamics 365 applications is integration with **Microsoft Teams**. This integration offers the following capabilities:

- Opening a Teams channel and assigning a Dynamics 365 record or table view to it. This will add the main table form or the selected table view to the channel. The members of the team can work around that record or view it using chat, document management, and other options. The documents that are created in such a channel that is connected to Dynamics 365 will be available from the table form outside of the Microsoft Teams environment. It is required to have SharePoint integration enabled to have access to those documents.

- Adding a whole Power Apps app or Dynamics 365 app to the Teams environment, to a team, chat, or meeting.
- Using a Teams chat inside Dynamics 365.

This collaboration capability is illustrated in the following screenshot:

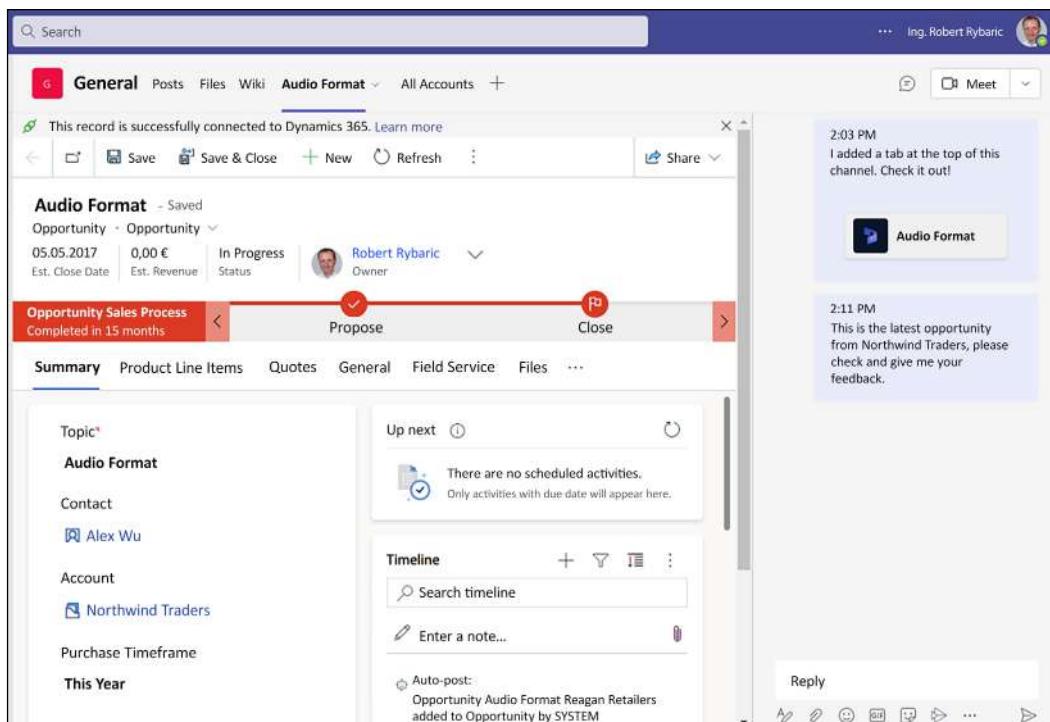


Figure 9.4: Integration with Microsoft Teams

As you can see, the integration with Teams is established, and in a team's channel a new tab was opened containing a Dynamics 365 opportunity. All the usual capabilities of Microsoft Teams, such as the chat or meeting capability, are available within this channel.

The integration configuration consists of the following simple steps:

1. Enable the integration in the settings in the Power Platform admin center.
2. Install and configure the Dynamics 365 app from within the Microsoft Teams environment.

After these simple installation steps, the user can work from within Microsoft Teams and use data from the integrated Dynamics 365 application as well as being able to start a chat directly from within Dynamics 365.

## Integrating with Microsoft OneNote

**Microsoft OneNote** is a note-taking application. OneNote can be integrated with Power Apps to provide all the benefits of OneNote from the context of a Power Apps record. It is possible to not just take text notes, but also use the drawing capability, as well as the audio and video recording capabilities of OneNote. OneNote integration requires that you have SharePoint integration enabled.

After enabling this integration, the capability is available from within the **Timeline** control on the table form, as illustrated in the following screenshot:

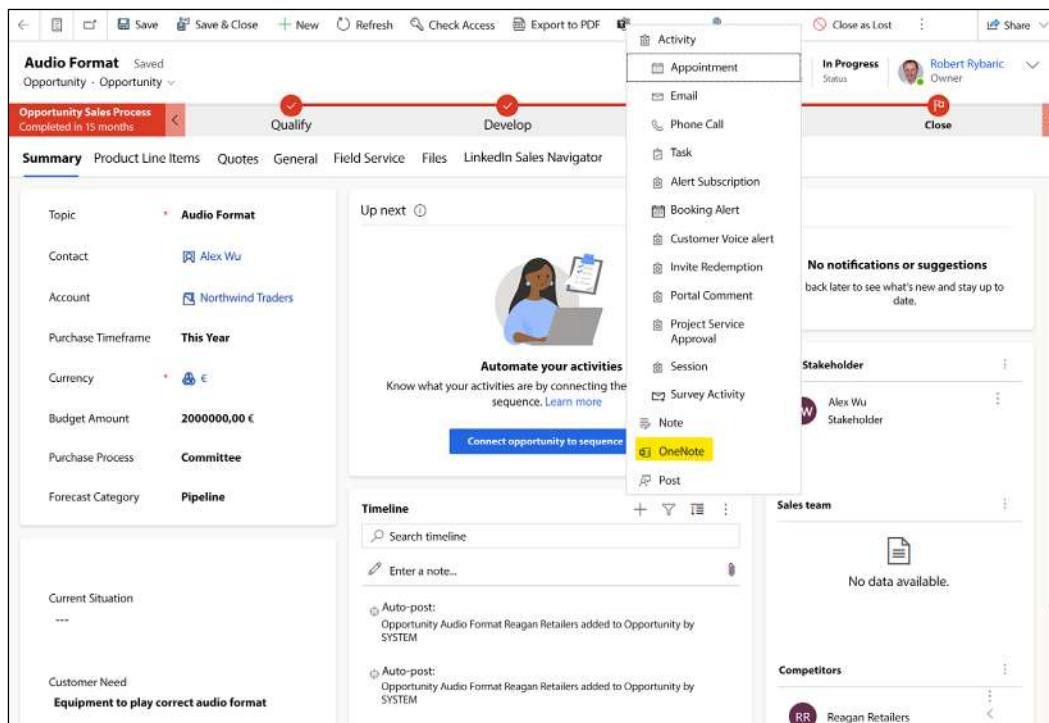


Figure 9.5: Integration with Microsoft OneNote

As we can see, we can open a OneNote file that was created for a particular record and use it to take notes, as well as using it with all the previously mentioned OneNote capabilities. The OneNote file is stored in the SharePoint folder that's attached to the Dataverse record.

The configuration steps for the OneNote integration process are as follows:

1. Turn on OneNote integration in the **Document Management** settings in the Power Platform admin center.
2. Select the tables to be enabled for the integration.

After following these simple configuration steps, integration will be enabled, and the capability will be available from the table records of the enabled tables.

## **Integrating with Skype/Skype for Business**

**Skype** (consumer) or **Skype for Business** can be integrated with model-driven Power Apps to provide a click-to-call capability for users. This capability makes it possible to trigger a phone call from within any Dataverse record containing telephone numbers. After clicking on the phone icon to the right of the **Telephone Number** column, the configured application (Skype or Skype for Business) will be ready to start the phone call. At the same time, a quick create form for a new **Phone Call** record will be opened for taking phone call notes.

Another interesting capability is the *Skype/Skype for Business presence bubble*, which will be displayed for all user records in the Dynamics 365 application.

This capability is illustrated in the following screenshot:

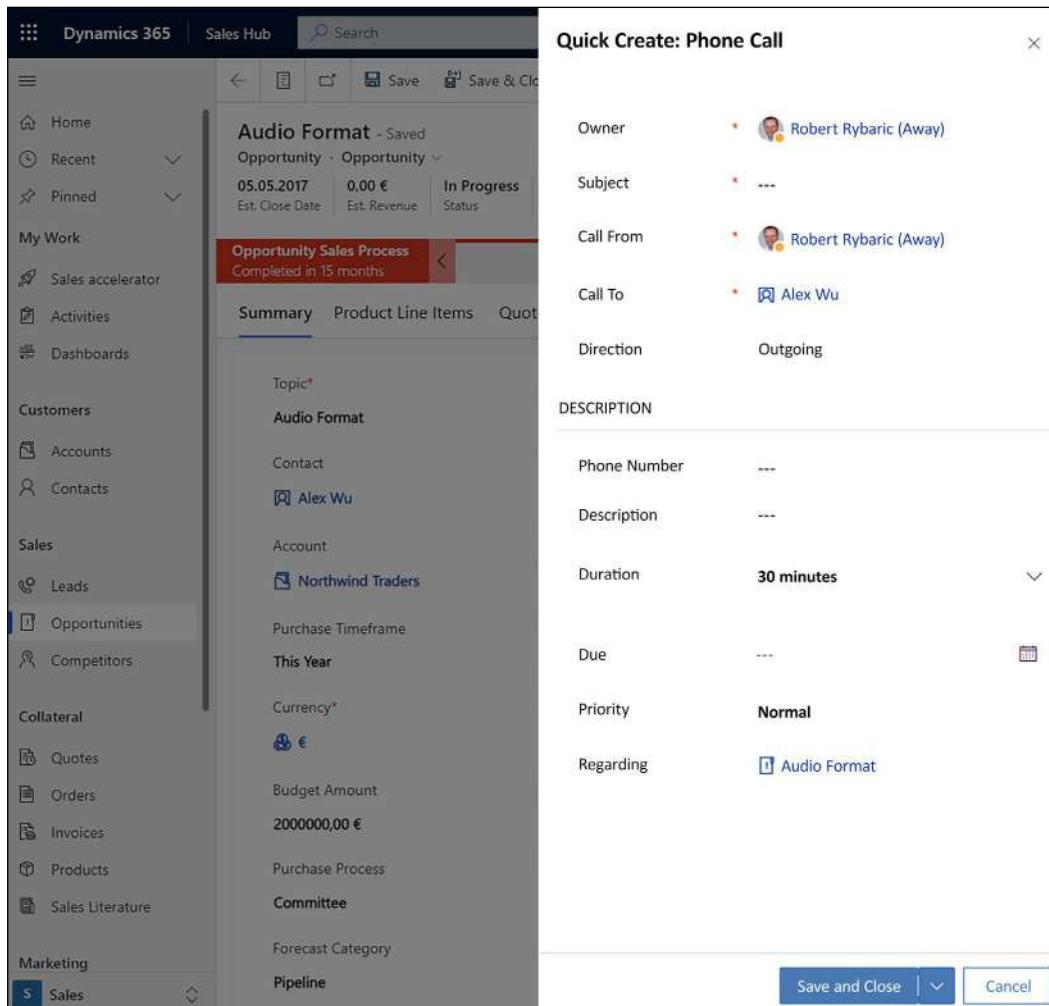


Figure 9.6: Dynamics 365 integration with Skype/Skype for Business

As you can see, by clicking on the **Business Phone** column's icon, the quick create form for the **Phone Call** activity opens, prepopulated with data from the record. The user record (shown in the **Owner** and **Call From** columns) displays the Skype for Business presence bubble.

To configure the integration, you need to enable the **presence functionality**, set up your **country/region code**, and choose **Skype or Skype for Business** in the settings.

## Integrations with Microsoft Azure services

Power Apps or Dynamics 365 applications can also integrate with various Microsoft Azure services by using *standard integration* capabilities. These integrations require either just some configuration steps, or in some cases, custom development.



We discussed some of the standard integrations in the previous chapters of this book. Here is an overview:

- **Azure Active Directory:** Azure Active Directory integration is used to provide authentication and/or authorization for all Microsoft cloud solutions, including Power Platform. For details, please refer to *Chapter 7, Microsoft Power Platform Security*.
- **Azure Service Bus:** Azure Service Bus integration can be used for remote code execution or integration with Dataverse applications. For details, please refer to *Chapter 8, Microsoft Power Platform Extensibility*.
- **Azure Event Hubs:** Azure Event Hubs integration can be used for outbound integration from Dataverse applications. For details, please refer to *Chapter 8, Microsoft Power Platform Extensibility*.
- **Webhooks:** This type of integration can also be used for outbound integration. The benefit of this is that you can use a synchronous communication pattern. For details, please refer to *Chapter 8, Microsoft Power Platform Extensibility*.

In the following sections, you will learn about some of the other, most important integration possibilities, leveraging standard integration.

## Integrating with Azure Blob Storage

Dataverse and Dynamics 365 applications can potentially store huge amounts of data in the attachments of annotations or email activities. Although the new storage model for Dataverse applications distinguishes between relational data and files, if the attachments are heavily used, they might occupy the available storage in the Dataverse database far too quickly. For such situations, you can completely offload all attachments from the business records to a dedicated **Azure blob**. Since Azure Blob Storage is much cheaper than Dataverse storage (or Dataverse capacity add-ons), it can be financially interesting to leverage this capability.

In order to enable this capability, follow these steps:

1. Use your own Microsoft Azure subscription to create a storage account with Azure Blob Storage.
2. Install one of the third-party storage management apps from Microsoft AppSource.



#### Important note

There was a Microsoft add-on available to cover this capability, but unfortunately, this add-on does not exist anymore. There are, however, multiple third-party add-ons that are able to offload the attachments to Azure Blob Storage, as well as to Dropbox or to SharePoint.

3. Configure the solution.



#### Important note

Offloading the attachments into Azure Blob storage will remove the offloaded content from the overall global search capability of the Dataverse platform.

After installing and configuring some of the solutions, all the file attachments will be automatically stored in Azure Blob Storage or any other of the possible storage types.

## Integrating with Azure Data Lake/Synapse Analytics

Dataverse offers a new capability to integrate with Azure Data Lake and Synapse Analytics in order to forward selected Dataverse data to an organization's storage and at the same time to offer the analytical capabilities of Azure Synapse Analytics.

This capability is built into the Power Apps Maker Portal, so that the configuration does not need any additional or third-party tools. What is required, however, is for an organization to have an Azure subscription with the following services provisioned:

- **Azure Storage account:** The storage account needs to be provisioned in the same Azure region as your Dataverse environment. In addition, the storage account needs to have the **Hierarchical namespace** feature switched on and the administrator needs to add the **Owner** role to the account.
- **Azure Synapse Analytics workspace:** The workspace needs to also be in the same region as the Dataverse environment. When configuring the workspace, the previously created storage account is used.



#### Important note

Azure Synapse Analytics is optional. The integration can be configured to the Azure Data Lake storage only, without the use of Synapse Analytics.

After these Azure services are fully provisioned and correctly configured, the Azure Synapse Link for Dataverse needs to be configured in the Power Apps Maker Portal. During the configuration, the Azure services created as prerequisites are entered, and afterwards, the required Dataverse tables are selected.



#### Important note

Only Dataverse tables with the activated track changes feature can be integrated.

As soon as the Azure Synapse Link for Dataverse is configured, the platform starts to replicate the data from Dataverse to Azure Storage (due to the **Hierarchical Namespace** setting, it is an Azure Data Lake storage).

The Azure Synapse Analytics service offers a cloud tool called Synapse Studio, which makes it possible to query the data in the data lake like a relational database, as illustrated in the following example:

The screenshot shows the Microsoft Azure Synapse Studio interface. On the left, there's a navigation pane with icons for Home, Data, Workspace, and Linked. Under 'Data', 'Lake database' is selected, showing 'dataverse\_production\_org930442c0'. Below it, 'Tables' are listed: account, contact, incident, lead, and opportunity. Under 'Views', there are none. In the center, a SQL script window titled 'SQL script 1' contains the following code:

```

1 SELECT TOP (100)
2 accountnumber, name, address1_line1, address1_city, address1_postalcode, telephone1
3 | FROM [dataverse_production_org930442c0].[dbo].[account]

```

Below the script, the 'Results' tab is active, showing a table with data from the account table. The columns are accountnumber, name, address1\_line1, address1\_city, address1\_postalcode, and telephone1. The data includes rows for Partner Sample, Adventure Works, Blue Yonder Air..., Alpine Ski House, City Power & Li..., Consolidated ..., Coho Winery, Fabrikam, Inc., and Litware. At the bottom of the results pane, a message says '00:00:01 Query executed successfully.'

Figure 9.7: Azure Synapse Studio

As you can see in the screenshot, there are records from the tables **account**, **contact**, **incident**, **lead**, and **opportunity** replicated to Azure Data Lake. With the help of Synapse Studio, the records can be retrieved using SQL statements as in the example for the **account** table.

After configuring the export, an initial synchronization will be performed, followed by continuous replication. The exported data will be available in Azure Storage in a collection of folders, where the data file for each table will be stored in a table-specific folder. In addition, in each table folder, there will be a sub-folder that contains a snapshot of the data, which is created at regular time intervals (by default, this is hourly). The purpose of this snapshot is to have a more stable copy of the Dataverse data, which is not constantly updated. The content of Azure Data Lake can also be viewed using *Microsoft Azure Storage Explorer*.

The data in Azure Data Lake can be used in various ways, for example, for building an analytical solution, or for integration purposes.

## Advanced integration scenarios with the Azure Synapse Link for Dataverse

The Azure Synapse Link for Dataverse opens a lot of possible advanced scenarios for analytics and integrations. In this section, we will briefly describe the solution approaches.

### Integration with Azure SQL or Azure Cosmos DB

One of the limitations of Dataverse is the inability to directly work with the underlying relational Azure SQL database that's holding the business data. Even though the latest TDS API allows *read-only access* to the underlying data, this capability cannot replace *full access* to a database, which is required in various analytical, data warehousing, and integration scenarios. In the past, there were solutions from Microsoft to replicate Dataverse data directly to Azure SQL Database or Azure Cosmos DB, but those solutions do not exist anymore. In this section, we will provide a design option for replicating data to the mentioned database systems using the Azure Synapse Link for Dataverse described in the previous section.

The first step in establishing the replication is to establish the Azure Synapse Link for Dataverse. After that, it is possible to make use of the **Azure Data Factory** service, which is a cloud ETL tool that is able to perform data integrations between various repositories. We can think of Azure Data Factory as a modern cloud version of SQL Server Integration Services. Let us now have a look at the practical implementation approach depicted in the following diagram:

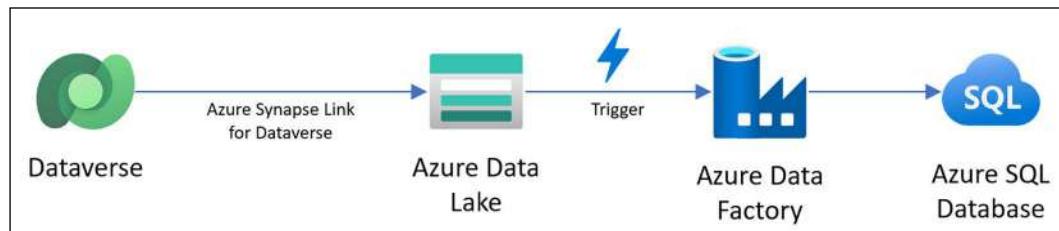


Figure 9.8: Integration with Azure SQL

As we can see in the diagram, the solution consists of the following components:

- **The Azure Synapse Link for Dataverse:** The first step in the integration is to enable the Azure Synapse Link to Dataverse, using the *Enable Incremental Update Folder Structure* setting.
- **Azure Data Lake storage:** The above configuration will start synchronizing selected data from Dataverse into Azure Data Lake.

- **The Azure Data Factory solution:** The solution consists of pipelines, a data flow, and a trigger, which starts the execution of the data transfer every time the Azure Synapse Link to Dataverse submits a new update of the Dataverse data.
- **Azure SQL Database:** The database is the target database, receiving the selected Dataverse data. Instead of Azure SQL Database, there can be any other database system, accessible from the Azure Data Factory service, for example Azure Cosmos DB.



#### Important note

Only Dataverse tables with the activated track changes feature can be integrated.

After installing and configuring the solution, the replication process will start working. This replication process runs in a near-real-time fashion (usually there is a refresh of the data every 15 minutes) so that the data in the target SQL database is always current and can be used for data integration, data warehousing, and reporting and analytics purposes.

## The Apache Spark engine

Part of the Azure Synapse ecosystem is the Apache Spark engine, which can be used to transform and analyze data coming from Dataverse.



#### Important note

Apache Spark in Azure Synapse Analytics is a Microsoft implementation of the Apache Spark parallel processing frameworks for big data analytics. For more details about the Apache Spark product, please refer to the product documentation: <https://spark.apache.org/>. For details about the Microsoft Azure implementation, please refer to the product documentation: <https://learn.microsoft.com/en-us/azure/synapse-analytics/spark/apache-spark-overview>.

An example of the use of this technology is illustrated in the following screenshot:

```

Notebook 1
Run all Undo Publish Outline Attach to dataverse ...
Ready
[13]
1 %%pspark
2 df = spark.sql("SELECT name, estimatedvalue from dataverse_sandbox_ong1652ea6a.opportunity")
3 df.write.mode("overwrite").saveAsTable("default.opportunity_details")
✓ 4 sec - Command executed in 4 sec 49 ms by admin on 4:25:42 PM, 11/15/22
> Job execution Succeeded Spark 2 executors 8 cores View in monitoring Open Spark UI
...
[14]
1 %%sql
2 SELECT * FROM default.opportunity_details
✓ 1 sec - Command executed in 1 sec 141 ms by admin on 4:25:43 PM, 11/15/22
> Job execution Succeeded Spark 2 executors 8 cores View in monitoring Open Spark UI
View Table Chart Export results
1 Café BG-1 Grinder for Alpine Ski House
10 Airport XL Coffee Makers for Alpine Ski House
10 orders of Product SKU JJ202 (sample)
12 Café A-100 Automatic Espresso Machines for Fabr...

```

Figure 9.9: Transforming and analyzing with Apache Spark

As you can see in the above screenshot, there are two cells – in the first cell, a view is created including the selected columns of the opportunity table, and in the second cell, that view is executed and displayed in chart form. The views created with the above approach are permanently stored in the storage and can be reused later.

## Power BI

The Azure Synapse Link for Dataverse is also well suited to building analytical solutions with Power BI. To connect to the data in Azure Synapse Analytics, it is possible to use one of the existing Power BI connectors:

- Azure Synapse Analytics SQL
- Azure Synapse Analytics workspace

After connecting, the content is presented in the usual way and can be used to build the required analytical solution. The benefit of this solution is that the analytical solution is not connected directly to Dataverse (with the DirectQuery mode) and so does not generate additional performance load. In addition, Azure Synapse Analytics can be extended with data from other data sources and this data can be included in the analytical solution as well.

## Integrating with Azure Logic Apps

Azure Logic Apps is an automation and integration cloud service that's widely used for cross-application scenarios. Logic Apps is the foundation technology for Power Automate and it provides the same concept of data connectors and a graphical designer. However, there are important differences, which make the use of Logic Apps more suitable for enterprise integration scenarios. These differences are documented in the following comparison chart:

Power Automate	Logic Apps
<ul style="list-style-type: none"><li>• Does not need An Azure license</li><li>• Cannot easily integrate with Microsoft Azure services</li><li>• Native integration with canvas apps</li><li>• Provide options for human interactions</li><li>• Can be deployed with Power Platform solutions</li><li>• Approval process</li><li>• Provides an app for mobile devices</li></ul>	<ul style="list-style-type: none"><li>• Requires an Azure license</li><li>• Can be easily integrated with other Azure services to build complex solutions</li><li>• Enterprise scalability</li><li>• Can be developed using code with Visual Studio</li></ul>

Figure 9.10: Comparing Power Automate with Azure Logic Apps

As we can see, Power Automate is more suitable for automations and simple integration within the world of Power Automate or Office 365. Azure Logic Apps is more suitable for advanced integration scenarios, including additional integrated Azure components, and with better support for pro IT developers.

The support for Dataverse applications is given by the Dataverse data connector, which works the same way as it does for Power Automate.

### Important note



For Logic Apps, the only available Dataverse data connector is *Dataverse (legacy)*. The *Dataverse* connector is not available since it needs to be used from within Power Platform solutions.

## Integrating with Azure API Management

**Azure API Management** is a robust Azure service for building externally facing APIs that provide integration endpoints to internal cloud solutions for your own on-premises systems or for partners, customers, and so on. API Management can be used as a useful component for many integration scenarios with Power Platform and, specifically, Dataverse solutions. The benefits of doing this are as follows:

- API Management can be used to expose a defined subset of the Dataverse APIs for external applications or third parties.
- API Management can be used as part of a more complex integration solution to define and expose certain APIs for external applications or third parties.

In both cases, the role of API Management is to isolate the internal structure of the integrated applications and expose only those services that need to be exposed for external use. In addition, API Management can provide a full set of various authentication and security features to protect endpoints from malicious attacks.

A direct integration between API Management and a Dataverse solution to expose some of the Dataverse APIs can be established using the following steps:

1. Create an API Management instance in the Azure subscription.
2. Register an integration application in Azure Active Directory to obtain the *application ID* and *client secret*.
3. Create a new application user in Dataverse by using the credentials from the previous step and provide the user with the proper security roles.
4. Configure the API, the required operations, and the security settings for the API Management endpoint. It is important to set up an OAuth 2.0 type of authentication and configure it using the credentials from *step 2*.

After this configuration is finished, the exposed API can be directly used to trigger the exposed operation(s) in Dataverse. It is also possible to use API Management to protect the inbound part of the exposed API using various security settings.

That concludes this section, where you have learned about many different standard integration possibilities between Dataverse and Microsoft 365 or Microsoft Azure components and services. In the next section, we are going to discuss the typical *integration scenarios, patterns, and solution approaches* for integrating Dataverse applications on the frontend side, as well as on the backend, with third-party IT on-premises or cloud systems and solutions.

## Frontend integration patterns and solution approaches

In this section, we are going to discuss the most common frontend integration patterns for Dataverse applications. Frontend integration can be used for a variety of reasons, such as to embed some third-party content on a Dataverse user interface or vice versa, or to perform client-side requests to third-party solutions. It is even possible to use certain capabilities of the Power Platform components to fully control the behavior of legacy IT solutions on the client side. Let's start by describing the client-side embedding options.

### Embedding third-party content into Dataverse

Model-driven applications allow us to embed third-party content into Dataverse table forms and dashboards. In both cases, the integration is implemented using the *IFrame* approach. An **IFrame** is an area on the table form or dashboard that *URL-based* content can be embedded into. In the simplest case, this content is static and does not depend on the context of the table record, nor any other context. Using static content might be useful for embedding into dashboards, but for table forms, the requirement would mainly be to make the content somehow related to the table type and the currently opened record. There are various ways to enable context and pass the context to the third-party URL:

- **Standard:** The standard method is to configure the IFrame properties to automatically pass the context data. When selected, the URL, which is specified in the IFrame, will be extended with a group of context parameters containing the GUID of the Dataverse record, the table type, the internal name of the Dataverse environment, and the language code of the Dataverse environment and the user. The embedded third-party application would need to retrieve this data and find the proper context information. To retrieve the data, a simple `HttpRequest.QueryString` method can be used.
- **Custom:** If the standard context information described previously is not sufficient for the embedded third-party application, custom code would need to be used to establish the URL for the IFrame. This can be implemented using a *JavaScript event handler*, configured for some of the proper events. This event handler would need to read all the necessary values from the table form and configure the IFrame's URL properly.

An example of the standard IFrame configuration is documented in the following screenshot:

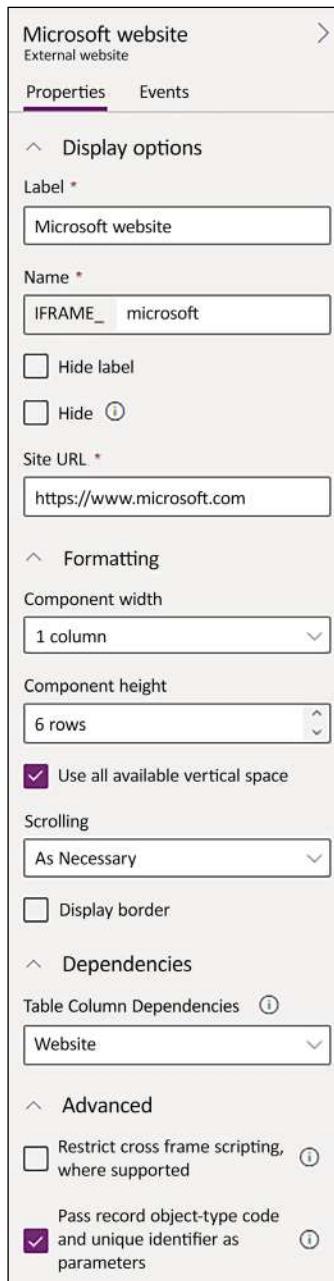


Figure 9.11: IFrame configuration for passing the standard context

As you can see, the configured URL is <https://www.microsoft.com> and the **Pass record object-type code and unique identifier as parameters** parameter has been checked. As a result of this setting, the URL that's passed to the IFrame will look as follows:

```
https://www.microsoft.com?id=90f1fa99-3788-e911-a818-000d3ab2dff3&typename  
=account&type=1&orgname=org6d9723d9&OrgLCID=1033&UserLCID=1033
```

As we can see, the *record ID*, *table type name* and *type code*, the internal Dataverse *environment name*, as well as the *language code* of the environment and of the user are passed as parameters.

There is another important parameter here, called **Restrict cross frame scripting, where supported**. When checked, this setting can limit some of the possibilities of the embedded application, such as using browser plug-ins, scripting, and more.

When using the *custom approach*, the URL would need to be formatted by the JavaScript event handler in a similar way, just using the required parameters.

## Embedding Dataverse content into third-party containers

It is sometimes required to embed parts of a Dataverse application into other third-party applications, such as websites, SharePoint sites, and Intranet portals. The reason for this can be to just publish certain important information from the Dataverse application in a central place that's visible to every user. These requirements can be addressed using the URL-addressable Dataverse resources. Dataverse allows us to address the following application elements with a specific URL:

- **Table forms:** It is possible to embed an empty table form, or a form containing data, from a selected record. If the table has more than one main form, it is also possible to specify which form to embed. It is also possible to specify whether the navigation elements should be visible in the embedded form.
- **Table views:** It is possible to embed system or personal views. For views, it is also possible to specify the visibility of the navigation elements.
- **Reports:** It is also possible to embed any of the SQL Server Reporting Services reports that are available in the Dataverse solution. In the URL, you can specify whether the report will be executed directly or whether there will firstly be offered the possibility to configure the filtering for the report.

It is important to understand that embedding Dataverse content does not automatically provide access to that content to everybody, but just for users with the proper Power Apps or Dynamics 365 license.

## Event-driven and on-demand frontend integration

You learned about the forms scripting and command bar extensibility capabilities of Dataverse solutions in *Chapter 8, Microsoft Power Platform Extensibility*. JavaScript event handlers can not only manipulate the content of Dataverse table forms but also implement communication with web service endpoints. They can call the Dataverse client-side Web API, but also any external web service. There are several popular methods for calling external web services from JavaScript code, including the following:

- **Ajax:** Ajax was traditionally used to make server-side calls from Dataverse JavaScript event handlers. The main method to trigger a server-side call with Ajax is the `XMLHttpRequest()` method. Depending on the type of web service endpoint, a possible authentication must be implemented and some of the basic HTTP calls, such as `GET`, `POST`, `PATCH`, and `DELETE`, must be used to retrieve data, create, update, or delete data in the target system, respectively.
- **jQuery:** Another approach to making server-side calls is to use the jQuery JavaScript library. jQuery offers several possible methods to make a server-side call, such as `$.ajax`, `$.get`, `$.post`, and so on. Again, the exact use of jQuery heavily depends on the type of external web service endpoint being used.

There are certainly some other possibilities since the JavaScript language has a huge expert community and there are many popular JavaScript libraries out there, making the implementation of this type of communication easy for developers.

The described possibilities for integration with external systems can be used for the event-driven pattern, as well as the on-demand pattern. The difference is that event-driven communication happens without any intervention from the end user, while on-demand communication must always be triggered by the end user.

It is possible to implement various integration scenarios using the frontend approach, some of which are as follows:

- Send data from a table form to an external web service endpoint for processing.
- Retrieve data from an external web service endpoint and fill in some table form controls with the data.
- Retrieve some data from the Dataverse application using the *client-side Web API*, combine this data with some data from the table form, and send it all to an external web service endpoint.

In the next section, you will learn about the **Unified Service Desk (USD)** client, which can help you implement frontend integration.

## Using Unified Service Desk

Another component that's used for frontend integration is USD, as we mentioned earlier in this book. USD makes it possible to automate business processes on the frontend between various types of legacy applications. The product supports frontend integration for the following application types:

- Windows desktop applications (Win32, .NET Windows Forms, **Windows Presentation Foundation (WPF)**)
- Java applications
- Web applications (including Dataverse itself, Silverlight, and Java applets)
- Citrix-hosted applications

With the support of the USD component known as **Hosted Application Toolkit (HAT)**, it is possible to set or retrieve form field values, click on buttons, and perform other frontend automation processes.

In this section, we have covered some of the most widely used client-side integration patterns and technical possibilities. The next topic we will investigate in this chapter is the backend integration of Dataverse or Dynamics 365 applications with other IT systems.

## Backend integration patterns and solution approaches

In the following sections, you will learn about the most typical integration patterns used within Dataverse-based solutions. For every pattern, we will present the possible solution approaches while using the technological background information you have learned about in this and in the previous chapters of this book.

### The remote procedure call pattern

The **remote procedure call (RPC)** pattern is the simplest integration pattern, where one application directly calls the interface of another application without using any middleware. This pattern is usually synchronous and consists of a request-response pair. The called application must provide a suitable API to be used from the calling application. This pattern can be seen in the following diagram:

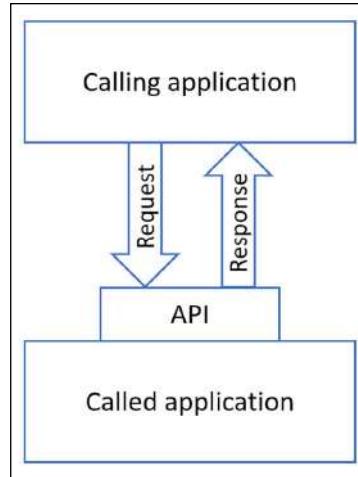


Figure 9.12: Remote procedure call pattern

As you can see, the calling application is directly calling the API of the called application with a request, and the API directly answers with a response.

For Dataverse solutions, there are two basic outbound implementation approaches, as shown in the following diagram:

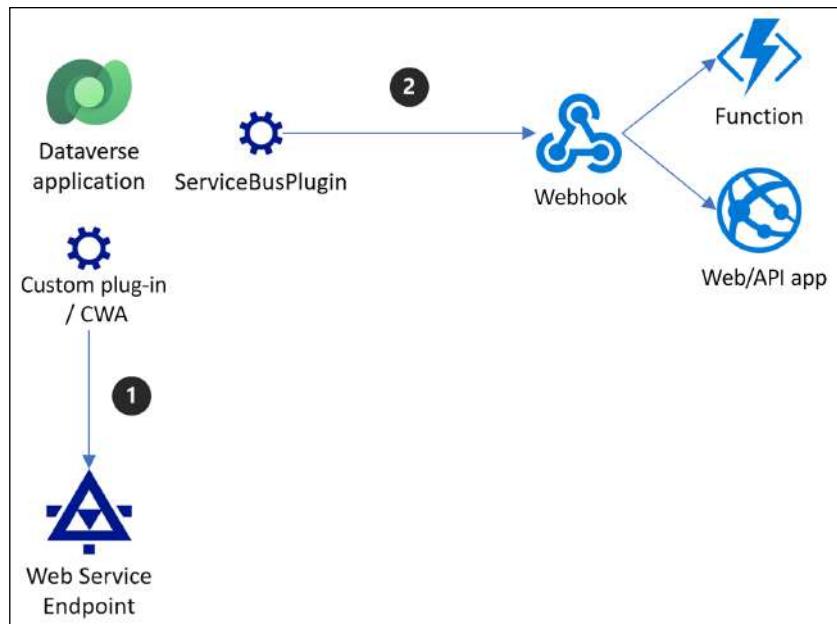


Figure 9.13: Outbound RPC solution approaches

As we can see, the following suitable implementation options are available:

1. **Custom plug-in or custom workflow action (CWA):** For this solution, it is required to develop a custom Dataverse plug-in or custom workflow action. The plug-in needs to be registered as synchronous or the workflow using the custom workflow action needs to be synchronous. The called service endpoint must be of the HTTP or HTTPS type with a DNS URL (no IP address) and must be deployed in the cloud. For on-premises deployments, it must be published on the internet. Azure API Management can be used to support building the on-premises endpoint as well. This solution can work with the possible response data coming from the service endpoint. The authentication limitations, the general timeout of 120 seconds, and the overall performance and scalability of this solution need to be considered.
2. **Webhook integration:** For this solution, the built-in plug-in for Azure integration can be used, but for integration with a webhook. The solution makes it possible to implement a synchronous communication pattern. The integrated endpoint has the same restrictions and considerations as the previous option. However, the solution approach does not support processing any response from the integrated service endpoint. Possible Microsoft cloud technologies to implement the webhook endpoint include Azure Functions, Azure Web Apps, and Azure API Apps.

The inbound implementation of this pattern is very simple, as shown in the following diagram:



Figure 9.14: Inbound RPC solution approach

As we can see, calling the native Dataverse API is basically an *inbound RPC solution* approach since the vast majority of the Dataverse API methods are synchronous and return an immediate response for every request. As you already learned in the previous chapters, the calling application can use the *Web API*, *SOAP*, or *TDS endpoints*. TDS endpoints can only be used to retrieve data. For every endpoint, the proper authentication needs to be implemented. The API limits that were described in *Chapter 3, Understanding the Microsoft Power Platform Architecture*, need to be considered.

## The relay pattern

With the **relay pattern**, one application is calling another application using a *middleware component*. The pattern can be synchronous or asynchronous and consists of a request-response, a pair, or a simple request (fire and forget). The called application must not provide a suitable API since this can be implemented in the middleware. This pattern can be illustrated with the following diagram:

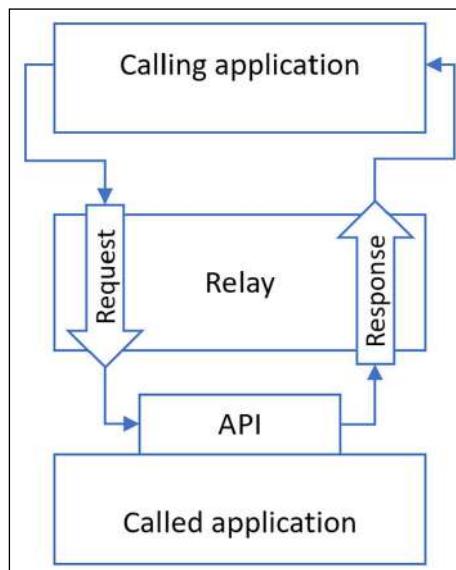


Figure 9.15: Relay pattern

As we can see, the **Calling application** is calling the **Relay** component, which forwards the call to the **Called application**. The same happens with the response. The benefits of this approach are being able to decouple the two applications and the possibility to implement some additional security, logging, tracing, and so on in the relay component.

For the outbound integration of Dataverse solutions, this pattern can be implemented using *Azure Service Bus integration*, as shown in the following diagram:

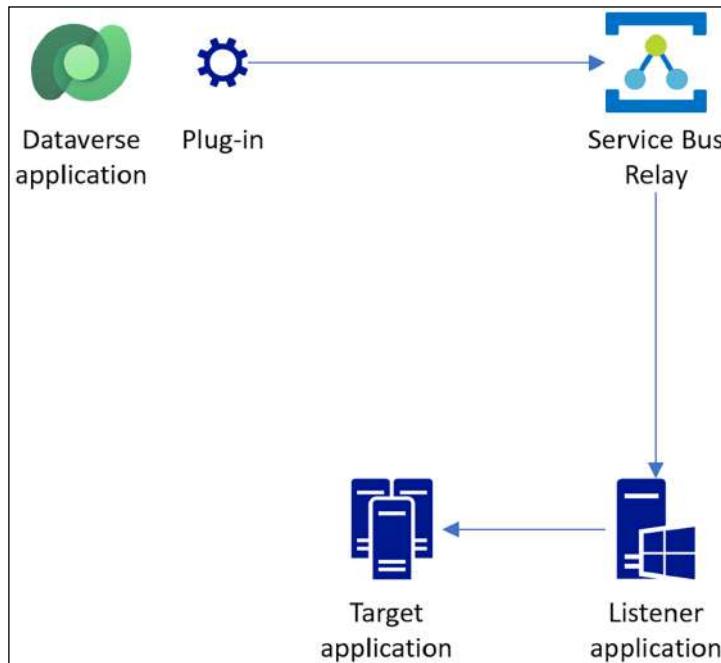


Figure 9.16: Outbound relay solution approach

As we can see, the solution consists of the Dataverse application integrated with Azure Service Bus in the relay mode and a **Listener application** communicating with both **Azure Service Bus** and the **Target application**. The benefit of this approach is that the solution does not require opening an inbound connection from the cloud, since the connection is always opened from the listener as outbound. The listener application must always run permanently to be able to immediately respond to remote execution requests coming from the Dataverse plug-in.

There are three different ways to implement the plug-in:

- **Built-in plug-in (ServiceBusPlugin):** This solution does not require any custom development on the plug-in side; however, the communication can be only asynchronous, and the solution cannot process the possible return values coming from the listener as a response.
- **Azure-aware plug-in:** This solution requires developing an Azure-aware plug-in, which is a specific plug-in type that reuses the standard Azure integration capabilities. This solution makes it possible to implement a synchronous communication pattern and to process the return values coming from the listener.

- **Custom plug-in:** This is the most flexible solution as it makes it possible to implement synchronous communication and any required business logic in the plug-in. The solution requires that we implement all the logic and the connection to Azure Service Bus in the plug-in code.

For inbound relay integration, the API Management component can be used either as a single component or in combination with additional cloud components, as illustrated in the following diagram:

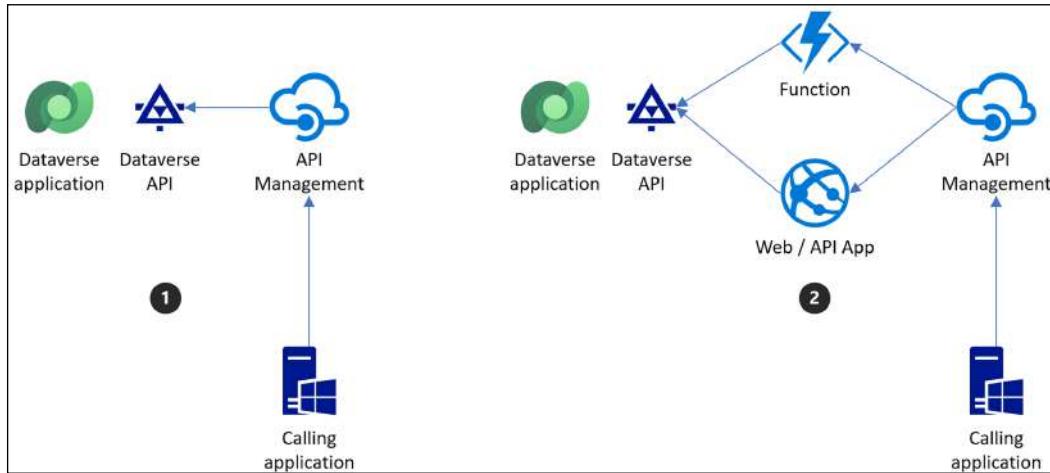


Figure 9.17: Inbound relay solution approach

As we can see, there are the following options:

- **API Management only:** For this solution, API Management is used to isolate the Dataverse API from the external integrated applications. The role of this component is to publish certain selected Dataverse API methods and to secure the communication channel using all the security features provided by API Management. However, this solution option cannot implement any data mapping or additional business logic.
- **API Management and additional components:** For this solution, the API Management component can be used in combination with additional cloud components such as *Azure Functions*, *Azure Web Apps*, or *Azure API Apps*. This solution approach adds the possibility of implementing any required mapping and business logic on top of the security features of API Management.

Since the *solution approach* represents synchronous communication, it is not recommended to use any asynchronously working component, such as Azure Logic Apps, but rather *components running synchronously*.

## The publish-subscribe pattern

The **publish-subscribe** pattern is a typical message-based pattern that works asynchronously and implements a *fire-and-forget approach* without a response to the communication. The pattern uses messaging middleware, as shown in the following diagram:

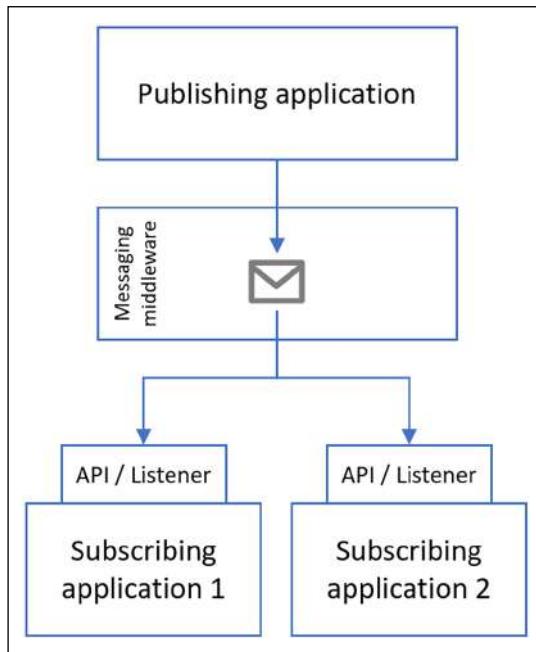


Figure 9.18: Publish-subscribe pattern

As we can see, the **Publishing application** is sending messages to the middleware, which acts as a pipeline distributing the messages to all subscribers.

There are two main solution approaches you can use to implement an outbound integration using the *publish-subscribe* pattern. The first approach is using the *default integration* with Azure Service Bus or Event Hubs, as illustrated in the following diagram:

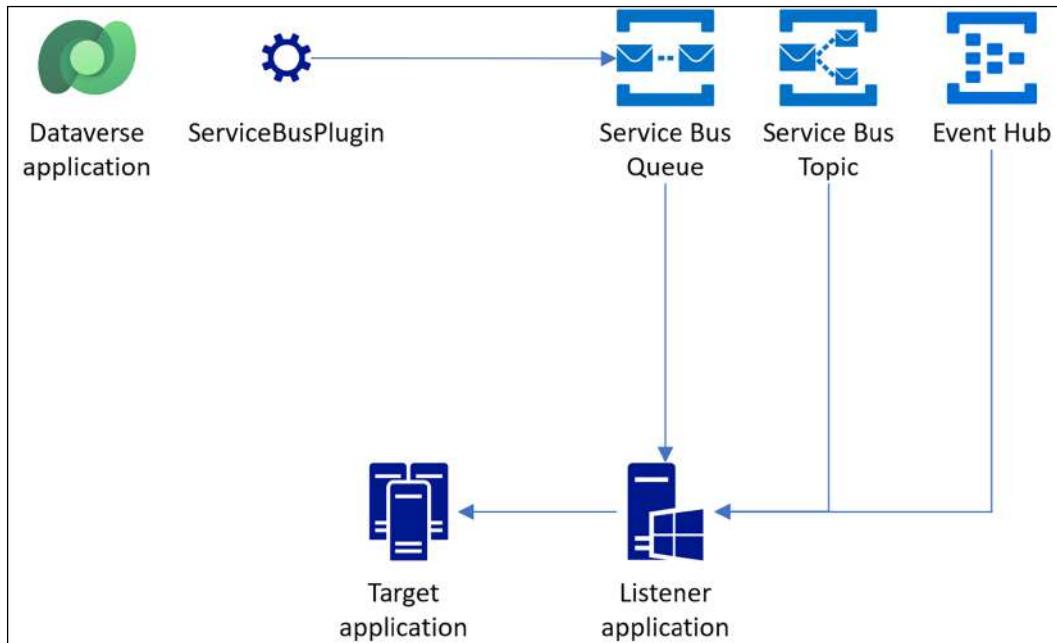


Figure 9.19: Outbound publish-subscribe solution approach 1

As we can see, the solution approach is using the built-in integration capabilities with **Azure Service Bus**, in this case not in the role of relay, but using the queue or topic components. The other possibility is to use the **Azure Event Hubs**. In all three variations, the integration on the Dataverse application side is implemented without the need for custom development; the only custom development efforts are to build the **Listener application**. Since this solution approach does not use the relay pattern, the Listener application does not need to run permanently, so the messages from Dataverse will wait until they are retrieved. The listener will receive the messages in the native Dataverse format and will need to implement all the necessary mapping, processing, and business logic.

The second approach is using *Azure Logic Apps* as an ideal component for building asynchronous integration solutions, as illustrated in the following diagram:

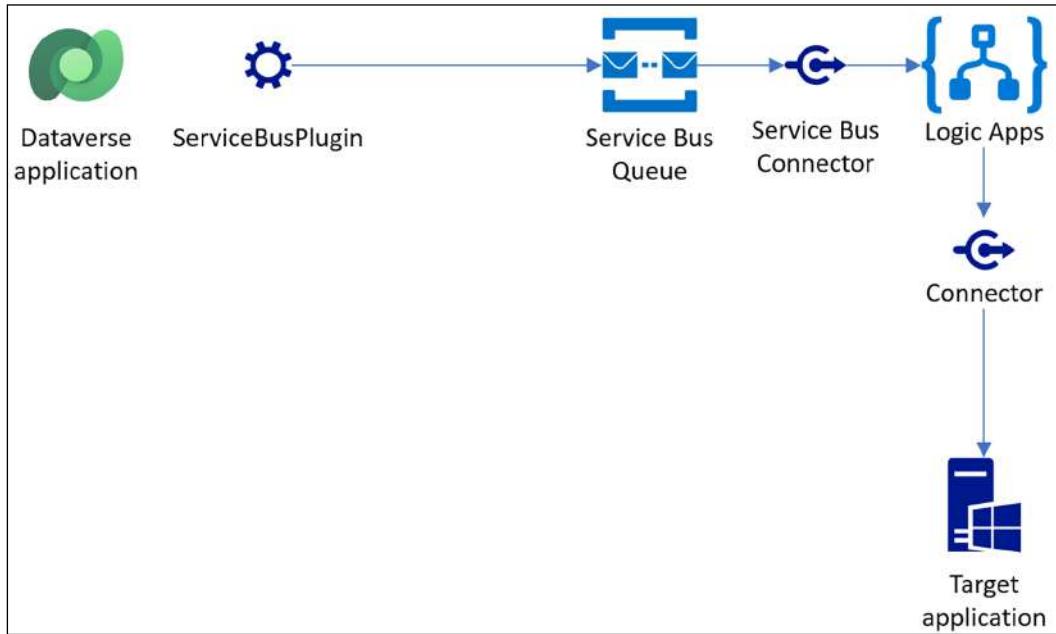


Figure 9.20: Outbound publish-subscribe solution approach 2

As we can see, the solution approach is still using the built-in integration with the **Azure Service Bus** component, but the whole business logic is implemented in the **Azure Logic Apps** component. The connector to the target application can be either any of the many public connectors or a custom connector, if the target application does not provide a supported API. This solution approach is asynchronous but near-real-time and provides high scalability.

Another variant of this solution approach is to completely avoid using Azure Service Bus and only use *Azure Logic Apps* with a Dataverse connector, as illustrated in the following diagram:

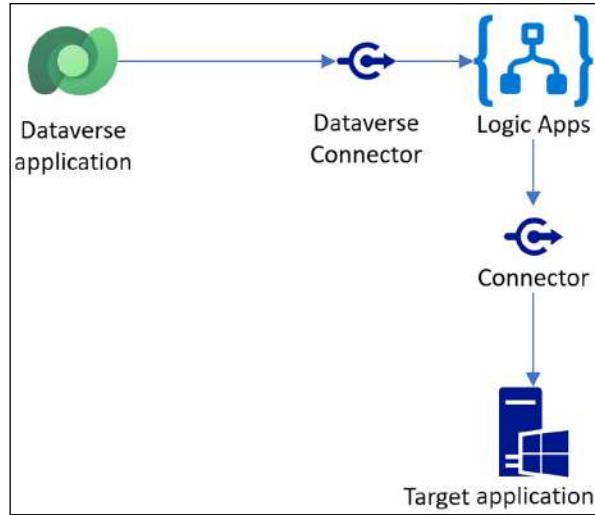


Figure 9.21: Outbound publish-subscribe solution approach 3

As we can see, this solution is implemented purely using **Azure Logic Apps**. This is a very simple solution and can be used if everything can be implemented in Azure Logic Apps. The solution is highly scalable and is asynchronous due to the asynchronous nature of the Logic Apps service, and also due to the capabilities of the Dataverse connector, which is a polling connector. This is also the only solution approach that can also be implemented using *Power Automate* instead of Logic Apps, since no other Azure cloud service is required.

An inbound publish-subscribe solution can be implemented by using *Azure API Management* combined with *Azure Logic Apps*, as shown in the following diagram:

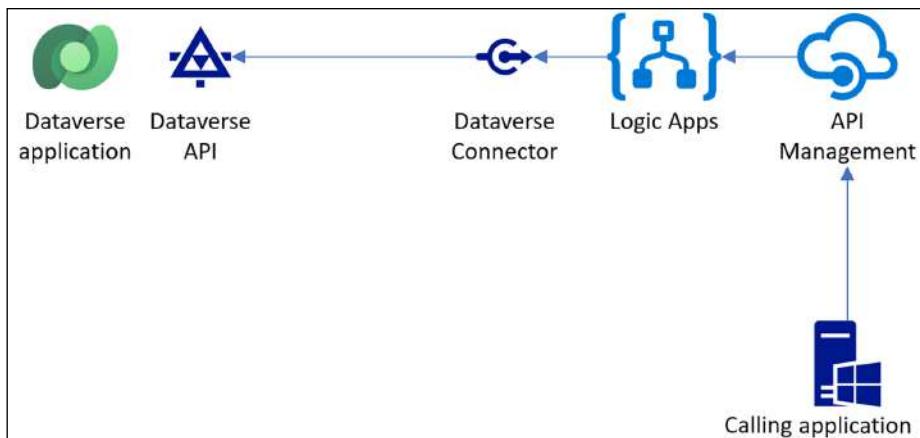


Figure 9.22: Inbound publish-subscribe solution approach

As we can see, the solution consists of the API Management components, which provide the externally facing API and the security layer, and Logic Apps, which provides all the necessary business logic. This solution is highly scalable and secure.

## The request-callback pattern

The **request-callback** pattern is an extension of the asynchronous *publish-subscribe* pattern as it extends this pattern with an asynchronous response, as shown in the following diagram:

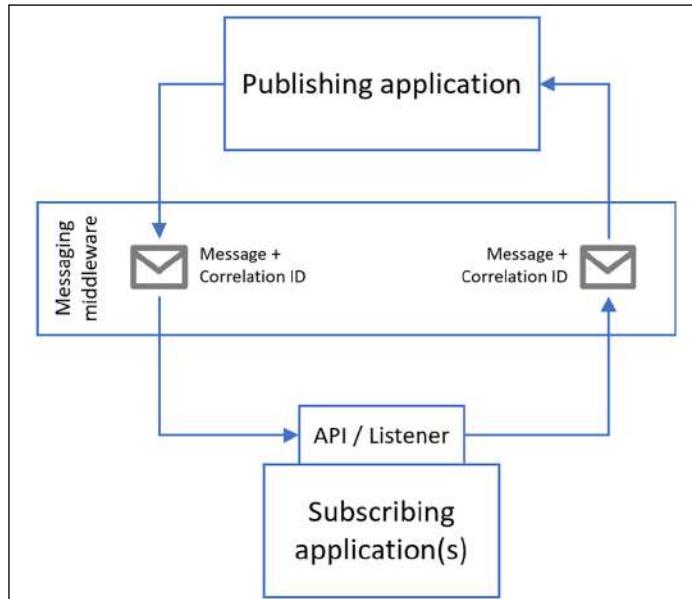


Figure 9.23: Request-callback pattern

As we can see, the publishing application is sending a message to the messaging middleware, where the subscribing applications can retrieve and process the message. The pattern requires the subscribing applications to send a response to the messaging middleware for the publishing application to acknowledge the message being processed. A very important aspect of this pattern is the use of a *correlation ID*, which must be identical in both the request and response so that the publishing application can assign the response to the right request.

For Dataverse applications, the patterns for outbound and inbound integration can be implemented as a combination of the solution approaches from the previous patterns. The following diagram shows the possible implementations:

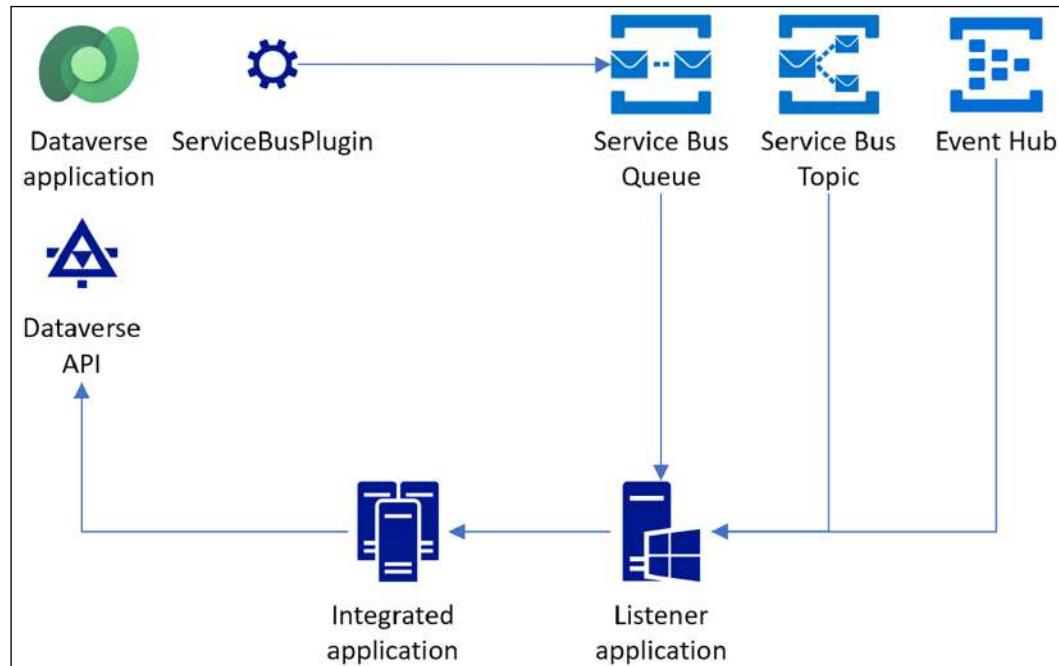


Figure 9.24: Outbound request-callback solution approach

As we can see, the *publish-subscribe approach* from the previous section was extended with the *response flow*, which was implemented by directly calling the Dataverse API. A variation of this solution approach would be to put an API Management component in front of the Dataverse API for increased protection.

Similar solution approaches can be designed using the other outbound publish-subscribe patterns by adding the proper *way-back* flow.

An inbound solution approach can be implemented using Azure Logic Apps in combination with Azure API Management, as illustrated in the following diagram:

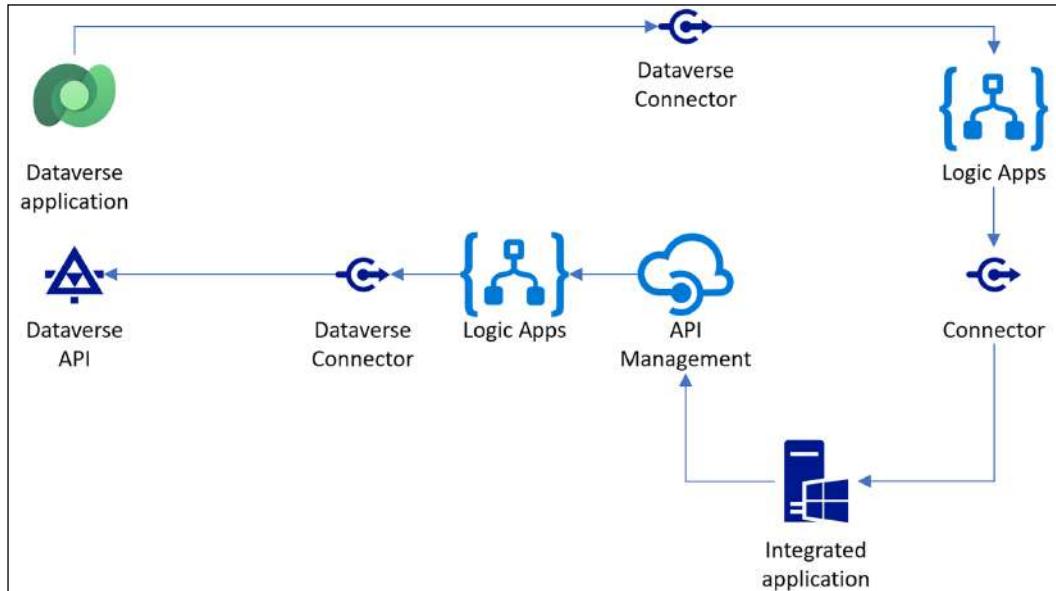


Figure 9.25: Inbound request-callback solution approach

As we can see, the solution consists of **Azure Logic Apps** components for implementing the whole outbound and inbound integration business logic, as well as the **API Management** component to secure the inbound API. The benefit of this solution approach is high scalability and security, and that Azure Logic Apps can easily implement the necessary correlation logic.

## Data integration

The last typical integration pattern is **data integration**. Compared to all the previous patterns, which are rather individual, real-time, or near-real-time, data integration is usually considered **mass integration**, where large amounts of data are synchronized at once using a *scheduling* approach. This pattern is illustrated in the following diagram:

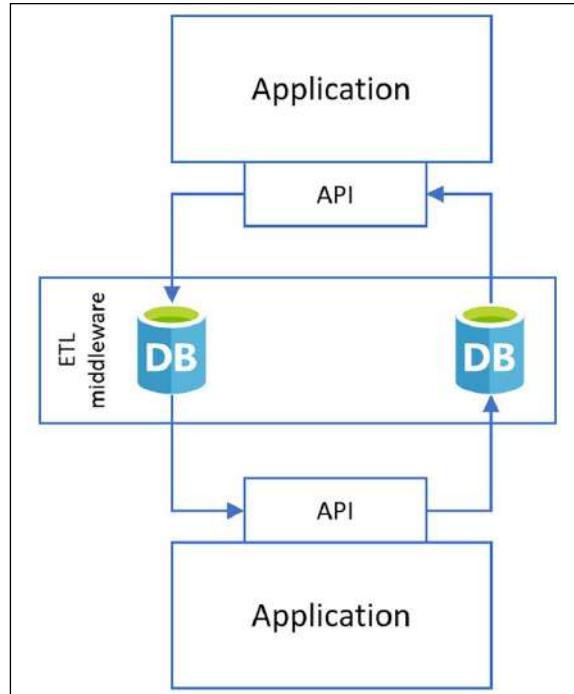


Figure 9.26: Data integration pattern

As we can see, the main component of this solution is an **extract-transform-load (ETL)** middleware, which is actively managing the integration processes. There are several integration approaches for Dataverse applications that follow this pattern. You will learn about some of those approaches in the subsequent sections.

## Virtual tables

Using **virtual tables** is a native Dataverse capability that allows you to include data from an external data source natively in the Dataverse application. It is not data integration in that sense, since there is no flow of data between the systems.

This capability is illustrated in the following diagram:

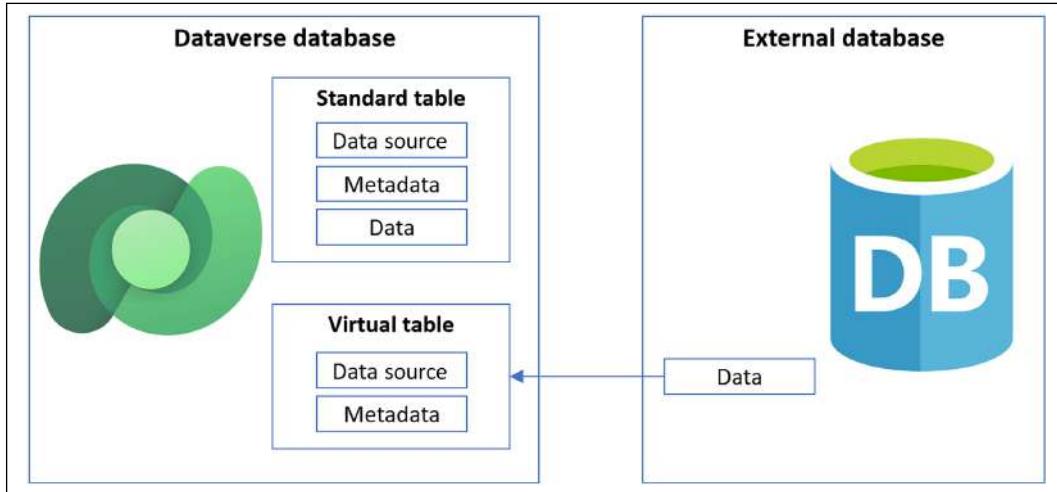


Figure 9.27: Virtual tables

As we can see, a standard table has the whole data source configuration, metadata, and the data itself stored in the Dataverse database directly. A virtual table has the data source configuration and metadata stored in Dataverse, but the business data resides in an external database. The purpose of this capability is to bring external data directly into the Dataverse application and make it part of the standard user interface, completely transparent for the end user. Virtual tables have several restrictions that must be considered when it comes to deciding whether to use this feature:

- The number of supported data types is limited. Certain Dataverse-specific data types, such as *Currency*, *Image*, and *Customer*, are not supported.
- Several table features are not supported, such as auditing, the activity type of the table, change tracking, offline capability, column security, and more.
- Virtual tables can only be organization-owned, not user/team-owned.

Virtual tables use the concept of **Data Provider Plug-ins**, which are used to make the connection between Dataverse and the external data source. Currently, the following plug-ins are available:

- **A standard OData 4.0 plug-in:** This plug-in is available by default from the Dataverse platform. This plug-in makes it possible to integrate with any external data source that supports an OData 4.0 web service endpoint.

- **Azure Cosmos DB plug-in:** This plug-in is dedicated to connecting to the Azure Cosmos DB cloud databases. By default, the plug-in is not installed in the Dataverse platform, so a separate installation from AppSource is necessary: [https://appsource.microsoft.com/en-us/product/dynamics-365/mscrm.documentdb\\_data\\_provider](https://appsource.microsoft.com/en-us/product/dynamics-365/mscrm.documentdb_data_provider).
- **Custom plug-in:** If neither of the existing Data Provider plug-ins is suitable to connect to a specific data source, you can develop a custom plug-in. This development process requires advanced developer skills, so it might be easier to develop an OData 4.0 interface for an existing data source rather than developing the custom plug-in.

Virtual tables can be customized in a similar way to standard tables, using the usual Dataverse customization tools.

#### Important note

At the time of writing this book, the virtual tables concept is being extended with a new preview capability of using Power Platform data connectors. This capability opens the door for many different external data sources to be integrated as visual tables into Dataverse. As of writing this book, the following Power Platform data connectors are supported: **SQL Server, Microsoft Excel Online (Business), Microsoft SharePoint**, and more are planned for the future.



In order to use this capability, a specific Microsoft solution, **Virtual connectors in Dataverse**, from AppSource needs to be installed first.

For more information, please refer to the product documentation: <https://learn.microsoft.com/en-us/power-apps/maker/data-platform/create-virtual-tables-using-connectors?tabs=sql>.

Using *virtual tables* is a very interesting approach since it has the benefit that the external data does not need to be physically replicated in Dataverse, but can reside in the original repository. Should the limitations of virtual tables be acceptable for the business scenario, then the concept is an excellent alternative to physical integration.

## Standard Azure data integrations

As we have learned in the previous sections of this chapter, there are certain standard data integration technologies for exporting Dataverse data into cloud services, such as Azure Data Lake and Azure Synapse Analytics, and in addition, also various SQL or NoSQL databases. All of these integration capabilities can be used to get a near-real-time copy of all or selected data from the Dataverse database. This data can be used for consolidation, data warehousing, analytics, and reporting, or further data integration with other IT solutions and IT systems.

## Integration between Dynamics 365 CRM and ERP

In *Chapter 1, Microsoft Power Platform and Microsoft Dynamics 365 Overview*, you learned about the various Dynamics 365 applications in regard to the CRM and ERP workloads. While they all belong under the Dynamics 365 umbrella, the underlying technology is very different since the main ERP modules – Dynamics 365 Finance and Dynamics 365 Supply Chain Management – are not based on the Dataverse technology. However, it is a legitimate expectation that the data from the CRM and ERP applications should be easily integrated. There is a solution we can use to configure integration streams between the two worlds called **dual-write**. With a properly configured dual-write solution, the following integration processes can be implemented:

- Synchronization of customer master
- Synchronization of product master
- Integrated business processes such as prospect-to-cash, procure-to-pay, project-to-cash, and more

The configuration of the dual-write solution must be performed from **Life Cycle Services (LCS)**, which is a service dedicated to the Dynamics 365 Finance and Supply Chain Management applications.

### Important note



For more information, please refer to the product documentation: <https://learn.microsoft.com/en-us/dynamics365/fin-ops-core/dev-itpro/data-entities/dual-write/dual-write-home-page>.

A similar solution exists for the integration between Dynamics 365 Business Central and Dataverse. Dynamics 365 Business Central provides a fully configurable connector to Dataverse, which makes it possible to use standard, pre-configured mappings or to modify them according to individual requirements.

**Important note**

For more information, please refer to the product documentation: <https://learn.microsoft.com/en-us/dynamics365/business-central/admin-common-data-service>.

Further details about these integrations are not in the scope of this book.

## Custom backend integrations

In this section, you learned about a lot of backend integration concepts that can be implemented using a *low-code/no-code* approach or simple custom development. However, it is possible, specifically for very complex integration requirements, to decide to implement the integration solution using *custom development* only, or to extend the solution with some custom code. As you learned in *Chapter 8, Microsoft Power Platform Extensibility*, Dataverse provides several API endpoints that can be used for implementing outbound as well as inbound integration solutions. As an example, the following are some possible custom integration solutions:

- Use the *TDS* endpoint of the Dataverse API to develop a batch outbound integration to retrieve larger datasets of Dataverse table data and use it to integrate with other IT solutions.
- Use the *Web API* or *SOAP*-based Dataverse API to implement any kind of outbound or inbound integration solution, by calling any of the CRUD methods available in that API.

With this, we have covered the typically used integration patterns and solution approaches. Now, we can go ahead with the remaining integration scenarios, including Power Virtual Agents, AI Builder, and Power BI.

## Other Power Platform integrations

In this section, you will learn some details about additional integration for other Power Platform cloud services such as Power Virtual Agent, AI Builder, and Power BI. Power Virtual Agent and AI Builder are components that need to be integrated, since they cannot work standalone. Power BI is a very flexible analytical solution with several integration possibilities.

### Power Virtual Agent and AI Builder

**Power Virtual Agent** and **AI Builder** are both Microsoft cloud services that provide the added value of a chatbot channel or artificial intelligence, integrated with business applications. Both products cannot work standalone, so they must be integrated with other cloud services.

*Power Virtual Agent*, as a chatbot solution, needs to be integrated into a public-facing environment, like a website or Microsoft Teams, to be available to users. The product supports standard integration into the following main channels:

- **Website:** The integration into a custom website can be achieved using a code snippet that's generated directly from the Power Virtual Agent designer.
- **Power Pages portals:** The chatbot can be integrated using a no-code control component, which can be integrated into a portal in the Power Pages studio.
- **Teams:** The chatbot can be integrated into the Teams environment using a simple customization of the Teams environment with Microsoft Teams App Studio.
- **Facebook:** The chatbot can be integrated into Facebook Messenger on an organization's Facebook page. The configuration of this integration requires creating a Facebook app in the Facebook development environment and integrating the chatbot with this app. The procedure is non-trivial and requires proper skills with the Facebook development environment. The final app must be approved by Facebook before it can be published.
- **Mobile app:** The chatbot can be integrated with a mobile application. Both mobile web as well as native mobile are supported. Integration with web mobile apps is simple. The same approach as for websites can be used. The integration with native mobile apps requires custom development changes on the mobile app side.

There are other integration possibilities for Power Virtual Agents chats, but they go beyond the scope of this book.

#### Important note



For further details about the integration possibilities of Power Virtual Agents, please refer to the product documentation: <https://docs.microsoft.com/en-us/power-virtual-agents/publication-fundamentals-publish-channels>.

*AI Builder* is used to easily build, train, and integrate AI models into other business applications. An existing AI model can be integrated in the following ways:

- **Power Automate:** Using Power Automate to integrate AI models can enhance any automation solution with the capabilities of the created model. Power Automate offers AI Builder actions for all of the custom or prebuilt models.

- **Power Apps:** Various AI models can be integrated into canvas apps, as well as model-driven apps. This integration can be implemented using the well-known customization tools *Power Apps Maker Portal* and *Power Apps Studio*.

In the next section, we will have a look at the Power BI integration possibilities.

## Power BI

Power BI visualizations can be integrated on the frontend with several Power Platform components. On the other hand, Power BI data can be integrated with various data sources in the backend. These capabilities make Power BI an ideal product to enhance existing solutions with analytics and reporting features.

Power BI visualizations can be integrated into the following Power Platform solutions:

- **Model-driven application:** Power BI tiles can be added to model-driven personal dashboards. At the same time, a whole Power BI report or dashboard can be included in a model-driven app as a system or personal dashboard.
- **Canvas apps:** Power BI tiles can be added to a canvas app and vice versa, canvas apps can be integrated into Power BI reports. This integration can be performed directly in Power Apps Studio, or in Power BI Desktop.
- **Power Pages portals/public website:** Power BI reports and its dashboard can be added to Power Pages portal web pages once this capability is enabled in the portal settings. This integration can be performed directly in the Power Pages Studio. Power BI elements can also be published on any public websites. In this case, everybody on the internet can see the Power BI content. No specific Power BI license is required for this type of integration.
- **Other cloud solutions:** Power BI content can be integrated with other cloud solutions, such as Teams or many popular solutions from third parties.

On the server side, Power BI offers an integration capability called **Power BI dataflows**. This capability makes it possible to define a data model. Every table in the data model connects to a physical data source. Part of the configuration is to select the required fields and perform some mapping, if necessary. A refresh frequency can be configured to specify the automated refresh rate of the data in the Power BI repository from the underlying data source. Once created, dataflows can be used in Power BI Desktop as data sources, to build the required visualizations.

With this, we have concluded the overview of the various integration capabilities of the Power Platform solutions. In the following sections, you will learn about some integration best practices for your Power Platform solutions.

## Learning about Power Platform integration best practices

In this section, we will present some best practices for frontend as well as for backend integration. Using some of the presented integration patterns and solution approaches is certainly useful, but not using them in the best way can cause various issues, especially regarding the solution's overall performance. Let's have a look at some of the best practices.

### Frontend integration

When deciding on which frontend integrations to use, there are some important best practices that should be followed in order to achieve a robust, maintainable, and performant solution:

- When embedding third-party content into Dataverse table forms, consider placing it on separate form tabs, not on the primary tab, to avoid waiting for the content to load.
- While jQuery is a very handy JavaScript library, it is not recommended to use it to call the Dataverse API. Use the *client-side Web API* to make calls to Dataverse, and only use jQuery to call external web services.
- For any server-side calls, always consider using *asynchronous communication*. Developing asynchronous communication is more complex, but the result is a smooth end user experience without unexpected delays and the user interface freezing. Using the XMLHttpRequest() method in a synchronous way is already deprecated and will be removed in future versions of the main browsers.
- Whenever possible, prefer on-demand frontend integration over event-driven integration. When frontend integration is implementing communication with third-party IT systems, it is fully dependent on the availability and performance of the external endpoints. Having this communication triggered by the end user gives the user more control and avoids unexpected delays or other abnormal situations.

In the next section, you will learn about some backend integration best practices.

### Backend integration

Backend integration also offers several possibilities where leveraging some best practices can lead to simpler code and faster and more reliable solutions.

The following are the best backend integration best practices:

- All the common *update* and *delete* methods of the Dataverse API require the use of the *primary key (GUID)* of the table record to identify the record to be updated or deleted. However, in integration scenarios, the primary keys of the Dataverse records are, in most cases, unknown by the integrated third-party solution or system. A good way to overcome this limitation is to hold the known record identifiers from the integrated systems as columns in the Dataverse tables (for example, customer number, product number, or case number) and configure those columns as **alternate keys**. Using alternate keys in integration scenarios makes it possible to use these known identifiers of the records in the update or delete methods instead of the primary keys, and it also saves an additional retrieve request to find the primary key first.
- In data integration scenarios, there is often a situation where some of the records coming from the third-party IT solution are not available in Dataverse yet, but some are already stored in Dataverse and just need to be updated. In order to simplify the solution, there is now the possibility to use the **UPSERT** request, which is part of the Dataverse API. An UPSERT request can automatically detect whether a record already exists in Dataverse. For existing records, there will be an automatic UPDATE transaction, while for non-existent records, a CREATE transaction is executed.
- In order to ensure there's a certain level of transactional safety and to pack multiple individual data modification requests into a single technical API call, it is possible to use **batch requests** when calling the Dataverse API. A batch request can contain up to 1,000 individual requests.
- For pure outbound integration scenarios, consider using the new Dataverse API endpoint known as **Tabular Data Stream (TDS)**. At the time of writing, this endpoint can only be used for reading data from the Dataverse database, but skilled SQL developers will quickly find it very useful for designing complex queries against Dataverse using the SQL language. The TDS endpoint is also very useful for checking Dataverse data using *SQL Server Management Studio*, or it can be used for building Power BI reports and dashboards.

This is the last section of this chapter. Now, as always, we will have a look at our fictitious customer Contoso Inc. to see what their key takeaways are, as well as how the information from this chapter will help them prepare an integration design.

## **Contoso Inc. Power Platform integration design**

The project team for Contoso Inc. is happy as they now fully understand the integration capabilities of a Power Platform solution. They are now designing the integration solution to seamlessly integrate their solution into their complex IT landscape. As usual, they have made a series of design decisions to make the most of Power Platform and to stay compliant with their policies.

Let's have a look at the decisions they made.

### **Integration with Microsoft 365 and Microsoft Azure**

Since the entirety of Contoso Inc. is already running a centralized Microsoft 365 tenant, they are happy to be able to integrate all their Microsoft 365 cloud services with Power Platform. They have specifically made the following decisions:

- A standard, server-side synchronization with Exchange Online in the tenant will be configured.
- An integration with SharePoint with the use of the third-party permission replication solution will be established, as already analyzed and documented in the security design.
- A full integration with Teams will be established since the employees of Contoso Inc. use Teams on a daily basis.
- The Azure Synapse Link for Dataverse will be established to have the possibility of a replica of their business data for reporting, analytics, and possible data integration.

### **Frontend integration**

Contoso Inc. have analyzed the frontend integration capabilities of Dataverse and Dynamics 365 solutions, and they have decided to leave it up to the project team regarding whether to use the embedding and JavaScript-based integrations. The main criterion for using these capabilities is the best possible end user experience, so the solution must not negatively influence performance. Furthermore, the solution must not compromise the security of the integrated IT system in any way.

The Contoso Inc. project team was quite impressed by Power Automate Desktop capabilities. They have decided to use this component to integrate some of their oldest IT systems, which cannot be integrated in the usual way due to the lack of a useful API.

The decision about using USD for customer service will be made later due to the perceived high implementation complexity. Contoso Inc. has decided to understand the details of the solution and whether the efforts to implement an agent desktop are reasonable first.

## Backend integration

Contoso Inc. have a multitude of existing IT systems that are candidates for backend integration. After carefully analyzing their capabilities, they decided to use the following approaches:

- Data from all existing IT systems that does not need to physically reside in Power Platform, will be integrated using *virtual tables*.
- The preferred way for integrating data that must be physically in the Power Platform solution and can be integrated asynchronously is a solution based on Azure Logic Apps in combination with Azure API Management. The project team considers this the most scalable and secure solution for Contoso Inc. They have evaluated that their existing IT systems have APIs, and it seems that all of them can be integrated by using some of the existing public data connectors. For integrating on-premises IT systems, an on-premises data gateway will be implemented in the most secure way.
- If there are any requirements for real-time integrations, the preferred method will be to use a solution based on Azure Service Bus in the relay mode.
- Since the Power Platform solution at Contoso Inc. will consist of CRM and ERP modules from the Dynamics 365 product family, they decided to further investigate the dual-write and virtual tables capabilities and to use them to the highest possible extent as a simple, reliable, and Microsoft-supported integration solution.

All of those high-level decisions will be documented in the Power Platform integration design document. Contoso Inc. is now confident that they have a good foundation to integrate the Power Platform solution into their complex IT landscape.

## Summary

This chapter was full of useful information for integrating a Power Platform solution into other Microsoft cloud services, as well as into existing IT systems and the solutions of an enterprise customer.

You have learned about the broad standard integration capabilities of Power Platform with Microsoft 365 and Microsoft Azure services. You have seen that many of the latest Dynamics 365 capabilities are no longer implemented in Power Platform itself but are instead integrated with some Microsoft cloud services. You have also learned about the different ways to perform frontend integration for certain Power Platform components. You now know that a backend integration can be implemented in a lot of possible ways and using a lot of various technologies, and that for every integration pattern, there are some useful solution approaches.

Finally, you became familiar with some proven integration best practices.

With this knowledge, you should be able to prepare a reliable integration strategy and design complex Power Platform solutions.

In the next and last chapter, we will focus on topics related to migrating data from various legacy data sources into a Power Platform solution.

# 10

## Microsoft Power Platform Data Migration

In this last chapter of this book, we are going to focus on the topic of **data migration** or **initial data load** as it's one of the most important and, for large projects, challenging parts of a Power Platform implementation.

When implementing a Power Platform solution, there is seldom the luxury to only deliver the solution itself, without taking into account the data that needs to be eventually migrated. So, it is an inevitable part of every large enterprise's implementation to identify all the possible data sources and design and implement a data migration solution. This chapter will help you better understand the specifics of migrating data into Power Platform, as well as the possibilities, approaches, tools, techniques, and challenges. Data migration always requires very good preparation, a detailed design, and most importantly – it requires a lot of time and effort.

In this chapter, we are going to cover the following main topics:

- Data migration overview
- Data migration tools and techniques
- Data migration challenges and best practices
- Contoso Inc. data migration design

## Contoso Inc. planning the data migration

Contoso Inc. is now almost at the end of creating the design for its Power Platform solution. They understand the security, extensibility, and integration aspects of a complex implementation. The last thing they need to focus on is ensuring all their existing valuable data in their various IT systems and solutions will be properly migrated to the Power Platform solution.

They understand that they need to analyze all the existing potential data sources and prepare the data in a certain way for migration. They now need to understand what the possibilities and options for preparing and migrating data are, what the usual challenges are, and how to overcome them to ensure the smooth transition of their solution into production.

## Getting an overview of data migration

**Data migration** into a Power Platform solution is one of the most important parts of an implementation project. However, it is often underestimated, and not enough time and resources are allocated for this task. It is really important to understand the importance of data migration, the possible approaches, and the tools and techniques that can be used to implement it. Without a well-thought-out plan and appropriate time and resources, the data migration can lead to last-minute project delivery stress and poor-quality migrated data.

In the following sections, you will learn about the different high-level approaches for importing data into a Dataverse-based application.

## Migration as part of integration

Often, a Power Platform solution is integrated with a lot of existing IT systems and solutions, some of which hold the required data that needs to be migrated into the solution. This is a fortunate situation since you may be able to reuse the integration solution for the data migration process.

In this case, it is better to call this approach **initial data load**, rather than data migration, since the source systems are not going to be shut down. Instead, they will continue to operate under new circumstances. The integration solution can be designed so that it is possible to trigger the initial data load process into Dataverse, as illustrated in the following diagram:

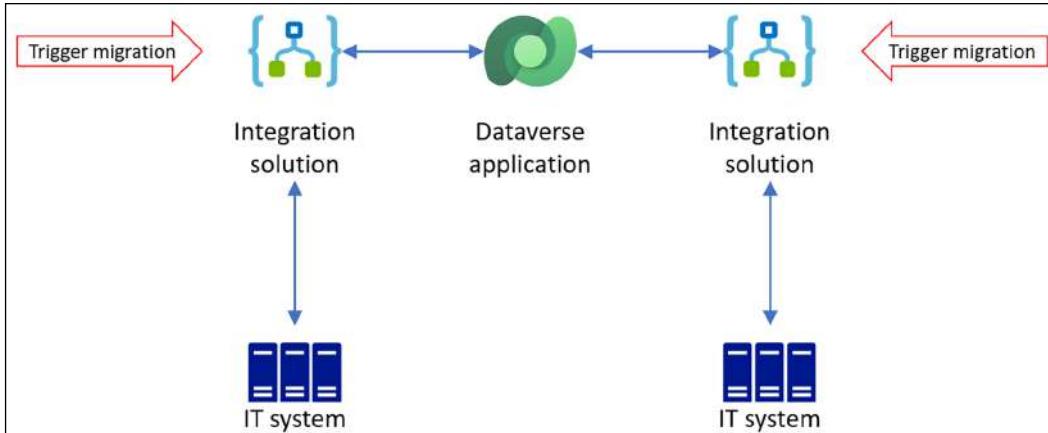


Figure 10.1: Migration as part of integration

As we can see, there are two *integrated* IT systems containing data that should be imported into the **Dataverse** solution at the beginning of the live operation. At the point of the initial data load, the integration solution is used to trigger the load into Dataverse as illustrated by the two arrows in the preceding diagram. This approach can be used if the following scenarios occur:

- The integrated IT systems continue to operate without any major changes.
- The data in the integrated IT systems is consistent enough that no upfront consolidation is required.
- There is a way for the integration solution to be reused for the data import process.
- This solution approach is compatible with possible real data migrations from other IT systems that will be shut down. It could be the case that the overall data from several IT systems needs to be mutually consolidated first, in which case this approach would not work. In that case, an *advanced migration* approach needs to be used – we will describe this later.

In the next section, we will describe another way to migrate data into Dataverse when the data structures and sources are simple and well consolidated.

## Migrating consolidated data

For smaller Power Platform solutions, where the number of potential data sources for migration is rather limited and the data is of good quality, we can migrate directly. A *direct* data migration, using some of the available technologies, can be done as in the following diagram:

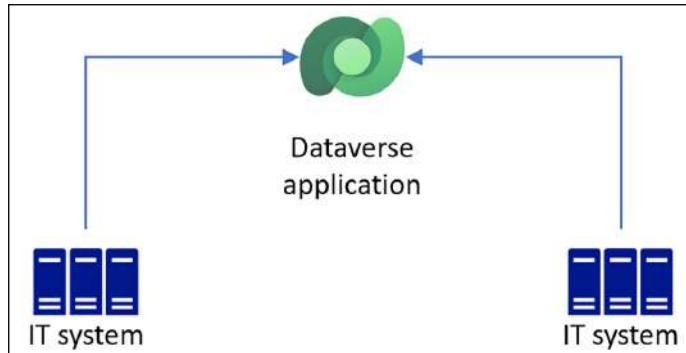


Figure 10.2: Migrating consolidated data

As we can see in the diagram, in this simplest case, a direct migration from multiple existing IT systems can be performed. This approach can be used, for instance, in the following situations:

- Every data source provides data from a distinct data domain (for example, one IT system can deliver the customer data, while another can deliver the product catalog data).
- The quality of the data is adequate for a direct import (no missing data in mandatory fields, proper mapping of fields possible, adequate relationship mappings possible, and so on).
- The data sources can provide data in a supported data format, such as Excel (XLSX), XML, CSV, and other formats, depending on the data import technology used.

However, in the majority of data migration scenarios, the circumstances are not so fortunate, and it is necessary to consider all sorts of complexities, inconsistencies, and poor data quality. As a result, only an advanced migration solution using a full **extract, transform, load (ETL)** approach can be used, as described in the following section.

## Advanced migration

In most cases, for large enterprise Power Platform projects, the beneficial conditions described in the preceding section simply do not apply and **advanced migration** needs to be used, as illustrated in the following diagram:

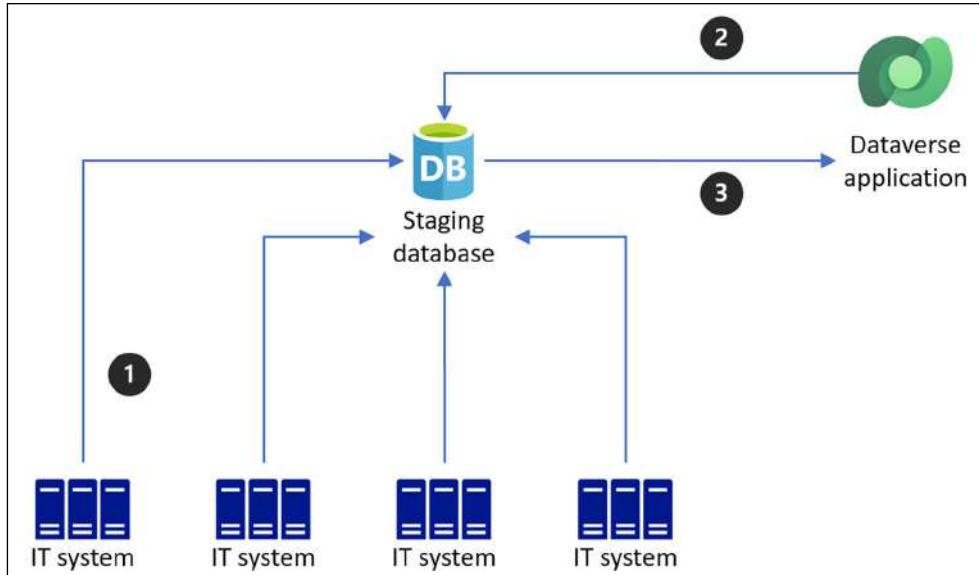


Figure 10.3: Advanced migration

As we can see, advanced data migration (using a typical ETL approach) consists of three major steps:

1. Loading all the required data from all data sources in all possible technological formats into a **Staging database**. The **Staging database** is the consolidation platform to consolidate the data, restore all the required relationships, cleanse the data of possible duplicates, and so on.
2. Enhance the data in the **Staging database** with Dataverse data to restore all the required references and relationships. This is especially important when part of the solution data is created directly in the Dataverse database and the mapping needs to be restored before the final migration. Another main reason can be to restore the ownership of records with real existing Dataverse users. In this step, the data structure in the **Staging database** must be prepared in a suitable form for direct import into Dataverse.
3. Directly load data into the Dataverse database. In this phase, the source data in the **Staging database** is already prepared and the data can be imported in the correct order. This helps with restoring all references and relationships.

Having understood how data migration works, in the next section, we will fully focus on the possible technology options available for migrating data into a Dataverse solution.

## Understanding the data migration tools and techniques

There are several ways to accomplish data migration to Dataverse. The Dataverse platform itself offers some out-of-the-box possibilities; however, they only cover less complex migration scenarios. For the most complex scenarios, an external solution needs to be considered, as you'll learn in the following sections.

### Entering data manually

For the smallest data volumes in which there are just a few records, the easiest and fastest way to migrate data is to manually enter it. Any semi-automated or automated data import would just take more effort and time to implement, compared to simple manual data entry.

### Using Excel files

The simplest way to import data into a Dataverse table is to use Excel's export feature. Dataverse applications allow you to export data from any table into Excel. There are several export possibilities, but for importing data, the **Static Worksheet** option needs to be used since it can be used for *reimporting* data back into Dataverse.

The following screenshot illustrates the export capability in any model-driven application:

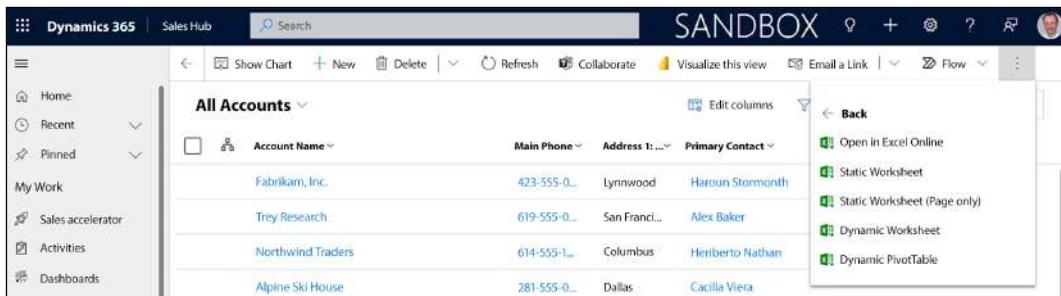


Figure 10.4: Export to Excel in model-driven apps

As you can see in this screenshot, in a model-driven app, there is always an option to export data into Excel files (as long as it is not prohibited by a security role setting). By using the **Static Worksheet** option, the exported file will contain information in the first three hidden columns, supporting the re-import of data.

Regardless of whether there is already some data in the table, this option can be used for both *importing* new or *updating* existing records.

The procedure to follow to import data using Excel files consists of the following steps:

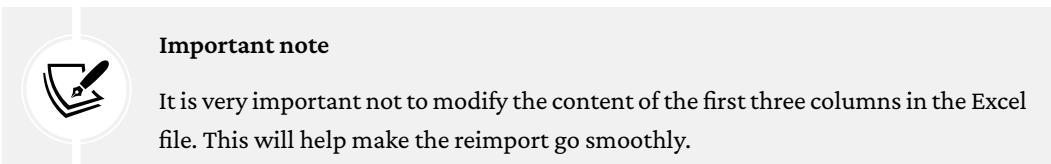
1. Prepare a system or personal view of the table, containing all the fields required for an import.
2. Select the prepared system or personal view in Dataverse and export data to Excel as a *static worksheet*. Note that the exported file will contain the same columns as the view used for export.
3. Manually enter all the required new data into the Excel file and/or modify the existing data in the file.
4. Import the data back into the Dataverse application.

An example of an Excel file is shown in the following screenshot:

A1	B1	C1	D1
1	(Do Not Modify) Account	(Do Not Modify) Row Checksum	(Do Not Modify) Account Name
2	83883308-7ad5-ea11-a813-000d3a33f3b4	vW5KigAABst8nA+mtE+6PUxXp08n	5/29/2021 12:24 A. Datum Corporation
3	7bf0c988-20bd-e911-a82e-000d3ab0842d	M2yPNWnnKt0KyMj8FwJ Era8UnsL	5/7/2021 12:25 A. Datum Corporation (sample)
4	6ff0c988-20bd-e911-a82e-000d3ab0842d	8A/hLCreyunXvdHr5jjFlp+wLH61Sn	8/18/2020 12:10 Adventure Works (sample)
5	81883308-7ad5-ea11-a813-000d3a33f3b4	xtiPsF2wRD2UIVUYCA4GMi5XBVYc	5/29/2021 12:24 Alpine Ski House
6	79f0c988-20bd-e911-a82e-000d3ab0842d	PDqcIkmkMQuLzvYisHmozeGxkr	10/24/2019 20:34 Alpine Ski House (sample)
7	73f0c988-20bd-e911-a82e-000d3ab0842d	ioqi+yMUYdZ0miHvgcILDQollmsTvN	10/24/2019 20:34 Blue Yonder Airlines (sample)
8	75f0c988-20bd-e911-a82e-000d3ab0842d	8qU/WKz54MKjLYakBOf7ah/OjbVr;	12/17/2020 13:18 City Power & Light (sample)
9	7df0c988-20bd-e911-a82e-000d3ab0842d	8wPgBmoartbaixaJWm3ZXukuMC	8/10/2020 12:24 Coho Winery (sample)
10	51c27fed-fad7-eb11-bac8-000d3addc84f	KrKlzlLzxuVfbKO9Gtw0pl6P+gSOUA	6/28/2021 21:36 Consolidated Sample
11	77f0c988-20bd-e911-a82e-000d3ab0842d	v6CQvTHz5yTd+rPz+Rbhifm4X2Cqc	3/4/2022 8:13 Contoso Pharmaceuticals (sample)
12	88cea450-cb0c-ea11-a813-000d3ab1223	ZgOzopQ76vU1tCBu2MPXueEB7EFT	3/28/2022 12:45 Fabrikam, Inc.
13	71f0c988-20bd-e911-a82e-000d3ab0842d	2calG5KKgZ9QqnqJZQ8dzsTyCSAt:	10/24/2019 20:34 Fabrikam, Inc. (sample)
14	6bf0c988-20bd-e911-a82e-000d3ab0842d	i1jUTnU1jKZj5L87YmKfsNsG65UYxJ	3/23/2021 10:44 Fourth Coffee (sample)
15	6df0c988-20bd-e911-a82e-000d3ab0842d	4f0+PGxVralGtExOe1zJ4kD963zcNi;	7/2/2020 11:35 Litware, Inc
16	b4cea450-cb0c-ea11-a813-000d3ab1223	ksYswdBj3/JdNRqfl4CTCBNbHb/M	6/8/2021 8:20 Northwind Traders
17	a4cea450-cb0c-ea11-a813-000d3ab1223	X/YIinK7IBQW5j4F/caiGizAx/q9t/4	5/29/2021 12:24 Trey Research

Figure 10.5: Exported Excel file from Dataverse

When opening the first three *hidden* columns, we can see the **ID** of the records, a **checksum**, and a **timestamp**. The content of the columns dictates the way the reimport happens: for new rows, there will be no ID in the first column, for modified rows, the checksum will be invalid.



The import and possible modification of data in a table will be as follows:

- You can add new rows to the file
- You can modify existing rows in the file

After you are done with the modifications, the next step is to reimport the file back to Dataverse, using a model-driven app again, according to the following screenshot:

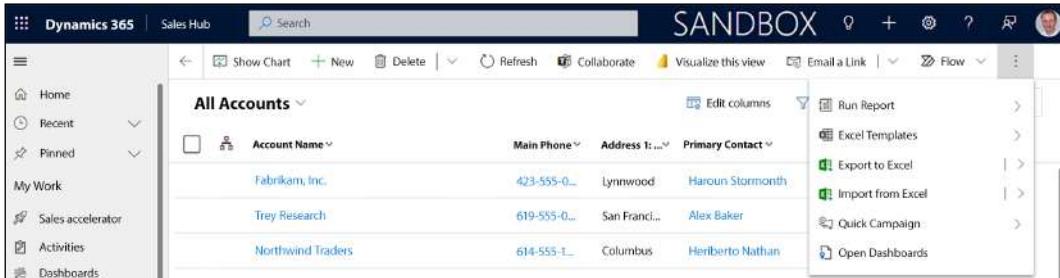


Figure 10.6: Import from Excel in model-driven apps

As you can see in the preceding screenshot, in model-driven apps, there is also the option to import data from Excel files into a selected Dataverse table. During the import of a previously exported file, the insert or update option for the rows will be recognized automatically.

Another way to use the Excel-based import feature is to download a *template* for the data import from the **Data Management** area in the Dataverse settings. These templates have the same structure as the template described for exporting to Excel but contain *all* the columns of the selected table, not just the selected view.

Importing data into Dataverse using Excel files is a good solution in the following situations:

- The amount of data to import is not very large (an Excel file cannot contain more than 1 million rows).
- It is possible to structure the data appropriately and it is good-quality.
- It is possible to fill in the foreign key fields (lookup fields) with the exact values of existing records to ensure existing relationships will be recreated.

Importing data using this approach can only be used to import into *one table*; importing into several tables within a single import step is not possible.

The next out-of-the-box data import option is the *data import wizard*, as described in the following section.

## Using the Dataverse data import wizard

If you require some more capabilities, such as importing data into several tables at once, you can use the **Dataverse data import wizard**. The wizard can be found under **Settings** in the Power Platform Admin Center.

This standard feature offers some more possibilities for importing data compared to the Excel import, such as the following:

- The following data formats are supported: XML Spreadsheet 2003 (XML), CSV, TXT, XLSX (Excel), and ZIP files.
- The wizard allows you to perform data mapping during the import process. The data maps that are created during the import process can be saved for later reuse. Alternatively, the data maps can be created upfront and imported into Dataverse for later import steps.
- The wizard allows you to import data into **several related tables** at once using the ZIP file option. In this case, data to be imported for Dataverse tables can be packed into a ZIP file and imported at once. During the import process, the columns relevant to existing data relationships can be specified. This makes it possible to recreate the relationships during the import process.

The following figure shows an example of the data mapping capability of the import wizard:

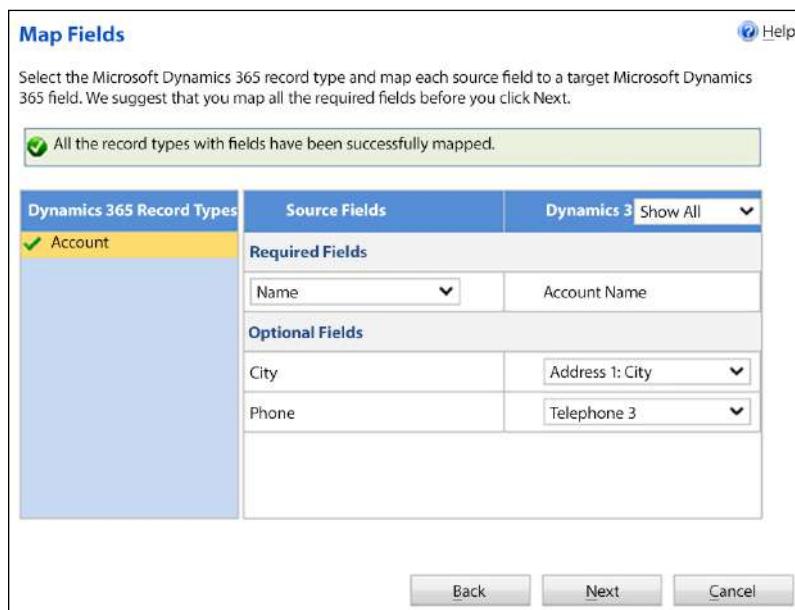


Figure 10.7: Field mapping in the import wizard

As we can see in the above part of the mapping area, there are **Required** table columns that must be mapped to columns from the import file. After that, the mapping of all other columns continues. In the last step of the import wizard, you can save the data map for future reuse.

The *import wizard* can be used to import small to medium-sized data files that have the appropriate structure so that they can be used for column mapping. There are the following file size limitations:

- Any file type containing data for a single table can have a maximum size of 8 MB. It is not recommended to import more than 20,000 records at once.
- When using a ZIP file to import data into multiple tables, the maximum file size is 32 MB.

The data import wizard's capability can also be used programmatically by using a collection of *Dataverse API* methods to invoke all the required import steps. The programmatic way offers additional possibilities for data mapping, including the following:

- Mapping transformations such as splitting data, concatenations, and replacing data
- Mapping owner information in the source data files to existing Dataverse users



Next, we'll have a more detailed look at the tool that's used for migrating small amounts of data from one Power Platform environment into another.

## The Configuration Migration Tool

You were briefly introduced to the **Configuration Migration Tool** in *Chapter 4, Power Platform Customization and Development Tools and Techniques*. In this section, we'll explain the capabilities of this tool in more detail with the help of the following diagram:

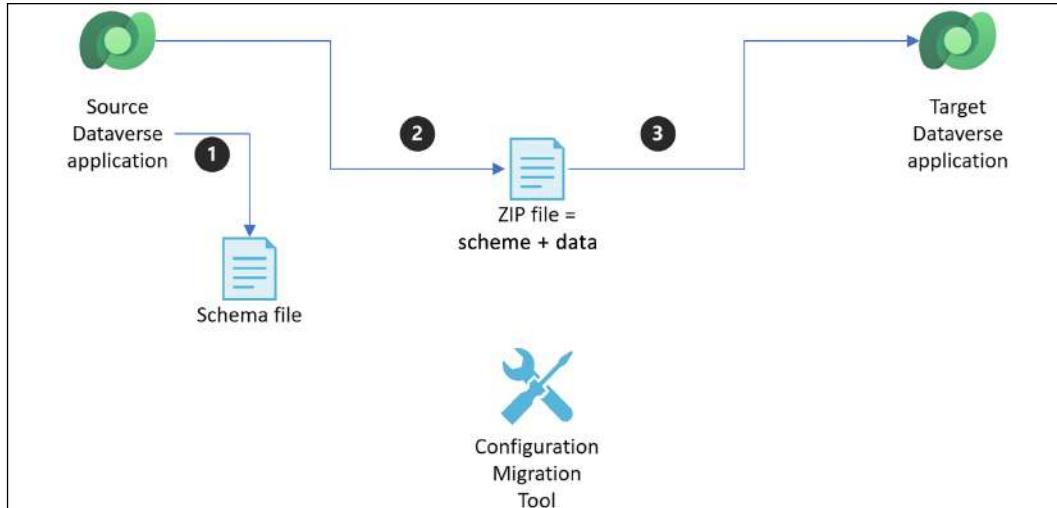


Figure 10.8: Capabilities of the Configuration Migration tool

As we can see, the tool works by following three steps:

1. In the first step, we select the **source** Dataverse environment and create the migration **Schema** by selecting all the tables and columns required for the migration. The **Schema** can be saved locally as an XML file. This file can be reused later for subsequent exports.
2. In the second step, the generated schema is used to create an export file that contains the schema, as well as the exported data in a ZIP file, which can again be saved locally.
3. In the third and last step, we can select the **target** Dataverse environment and use the exported ZIP file from the previous step to import the data into the environment.

The primary purpose of this tool is to migrate small, static configuration data chunks between environments to save time when it comes to manual data entry or migrating data table by table using the *Excel file* approach.

The tool has the following capabilities:

- Filter records for migration.
- Preserve relationships between data in migrated tables.
- Validate the created schema for consistency.
- Move the date-time field values forward. This can be used when preparing environments for demo presentations where the current datetime values are required.
- Map users from the source environment to users in the target environment.

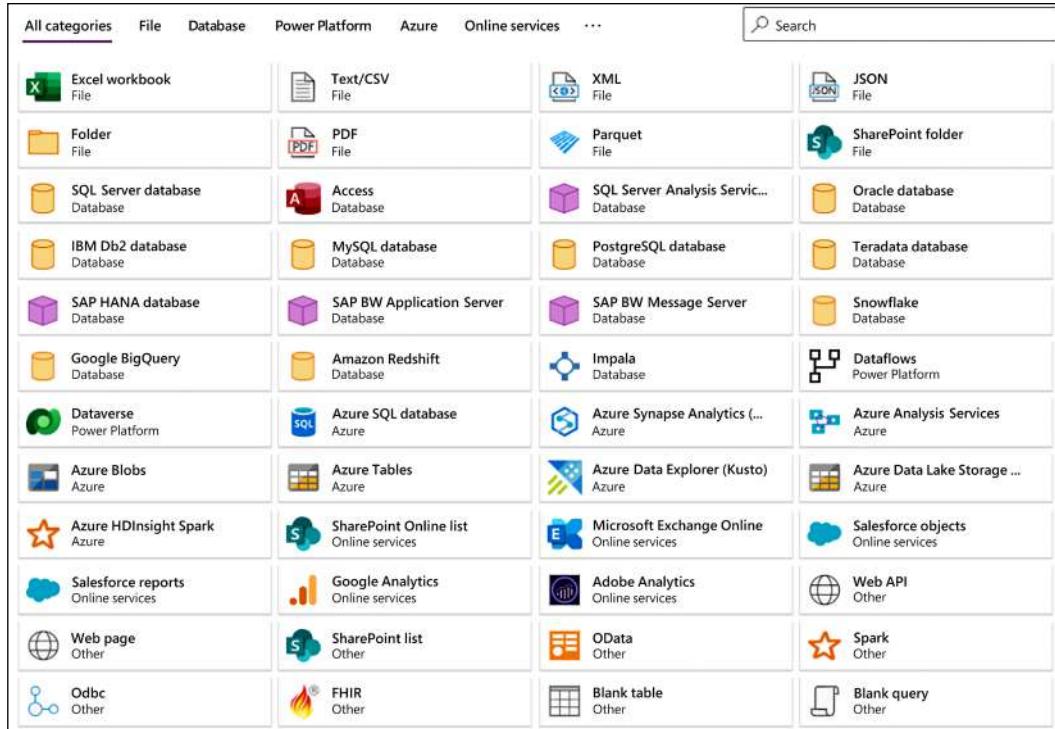
The tool can also be used for migrating small amounts of business data between Dataverse environments.

Another benefit of the Configuration Migration Tool is that you can include the exported data files in the **Package Deployer installation** package, as described in *Chapter 4, Power Platform Customization and Development Tools and Techniques*.

For more advanced data migration scenarios, we can consider using another Dataverse out-of-the-box capability known as *dataflows*, as explained in the following section.

## Dataflows with Power Query

**Dataflows** are an advanced data migration tool that's directly available in the Power Apps maker portal. Dataflows can import data from a large number of different data sources, as illustrated in the following screenshot:



The screenshot shows a user interface for selecting data import sources. At the top, there is a navigation bar with tabs: All categories, File, Database, Power Platform, Azure, Online services, and a '...' button. To the right of the tabs is a search bar with a magnifying glass icon. Below the navigation bar is a grid of 24 items, each representing a different data source with its corresponding icon and name. The items are arranged in four columns and six rows. The data sources listed are:

- Excel workbook File
- Text/CSV File
- XML File
- JSON File
- Folder File
- PDF File
- Parquet File
- SharePoint folder File
- SQL Server database Database
- Access Database
- SQL Server Analysis Servic... Database
- Oracle database Database
- IBM Db2 database Database
- MySQL database Database
- PostgreSQL database Database
- Teradata database Database
- SAP HANA database Database
- SAP BW Application Server Database
- SAP BW Message Server Database
- Snowflake Database
- Google BigQuery Database
- Amazon Redshift Database
- Impala Database
- Dataflows Power Platform
- Dataverse Power Platform
- Azure SQL database Azure
- Azure Synapse Analytics (...) Azure
- Azure Analysis Services Azure
- Azure Blobs Azure
- Azure Tables Azure
- Azure Data Explorer (Kusto) Azure
- Azure Data Lake Storage ... Azure
- Azure HDInsight Spark Azure
- SharePoint Online list Online services
- Microsoft Exchange Online Online services
- Salesforce objects Online services
- Salesforce reports Online services
- Google Analytics Online services
- Adobe Analytics Online services
- Web API Other
- Web page Other
- SharePoint list Other
- OData Other
- Spark Other
- OdBC Other
- FHIR Other
- Blank table Other
- Blank query Other

Figure 10.9: Dataflows data import sources

As we can see, many of the widely used database systems, business applications, and other data sources are supported, and the number of sources is growing constantly. The tool has the following capabilities:

- It is possible to specify the data to be imported for multiple tables.
- The tool offers advanced mapping and data transformation capabilities provided by the *Power Query* data transformation language.
- When importing data into Dataverse, the import can be performed for existing tables, as well as for tables created on the fly during the import.
- Relationships between imported data are preserved.
- The dataflow can be configured for one-time immediate execution or for scheduled execution. This capability makes it possible to implement a data integration solution that regularly refreshes the Dataverse data.

Dataflows can be used as the preferred tool for every situation where data is available in consolidated structures, ready to be imported into Dataverse after performing some transformations and mappings.



**Important note**

Currently, only a maximum of 500,000 records can be imported using a single dataflows project.

It is also important to understand that dataflows can only use cloud data sources by default. To use an on-premises data source, the *on-premises data gateway* product must be installed.

Dataflows can be also used to migrate data between two Dataverse environments. In this scenario, the dataflows are created in the target environment, and the Dataverse data source that's been configured with the URL of the source Dataverse environment needs to be selected.

With this, we have exhausted all the standard, out-of-the-box possibilities for importing data into a Power Platform solution. You have seen that there are various possibilities for simple to medium-complexity migrations. If these tools do not fulfill very complex migration requirements, there is a need to reach out to an external toolset, as explained in the following sections.

## SQL Server Integration Services

For the most complex data migration requirements, there is a need to use a robust and generic *ETL* tool that can perform any kind of data preparation before the actual migration happens. The preferred tool for the most complex Dataverse data migrations is **Microsoft SQL Server Integration Services (SSIS)**.

### Important note



It is possible to use an on-premises Microsoft SQL Server instance, or an instance deployed in the cloud, within **Azure Virtual Machine**. For very large data migrations, it could be useful to deploy the whole toolset into Microsoft Azure, to the same cloud region as Dataverse, to achieve the smallest possible latency time when communicating between SSIS and the Dataverse API.

To make the life of the data migration developer easier, it is recommended to use a proper connector to Dataverse. Since Microsoft does not provide a dedicated native Dataverse connector, it is necessary to use a third-party product. The recommended product is the *SSIS Integration Toolkit for Microsoft Dynamics 365* from the Microsoft partner company *KingswaySoft*. This connector toolset has gained general popularity and has been widely adopted by the expert community. The product has the capability to easily connect to Dataverse, as well as to the whole product family of Dynamics 365 CRM, Finance, Supply Chain Management, Commerce, Human Resources, and Business Central. The product comes with a free developer license. For production use, an appropriate commercial license is necessary.

With this toolset, every kind of complex data migration can be designed, developed, and executed. Since we are using the *Microsoft SQL Server* product, it is possible to use the database engine to create a staging database first. In this database, consolidation of the imported data can occur in its entirety.

A complex data migration using SSIS consists of all three typical ETL phases, as described in the following sections.

### Extracting data

In this first phase, all the data from all legacy data sources needs to be imported into the staging database. There are too many types of source data, so there is no single best way to perform this task. To import data from various major database systems, you can use the linked servers capability of Microsoft SQL Server.

Using this capability, you can connect a third-party database server and easily import all the required data into the staging database using SQL statements. Note, that first, you need to install a linked server **provider** for the respective database system into the SQL Server instance.

Importing data from many other data sources can be done using the capabilities of SSIS. This tool offers a broad range of data flow connectors to various technologies.

Part of the extraction phase can also be to import some of the existing data from the Dataverse database. This helps us perform the proper mapping between the data from the sources and Dataverse. Using the new TDS endpoint makes it easier for the developer to work with the Dataverse database. It is possible to open both the staging database and the Dataverse database in the **Microsoft SQL Server Management Studio** in parallel, in order to work with the data, as illustrated in the following screenshot:

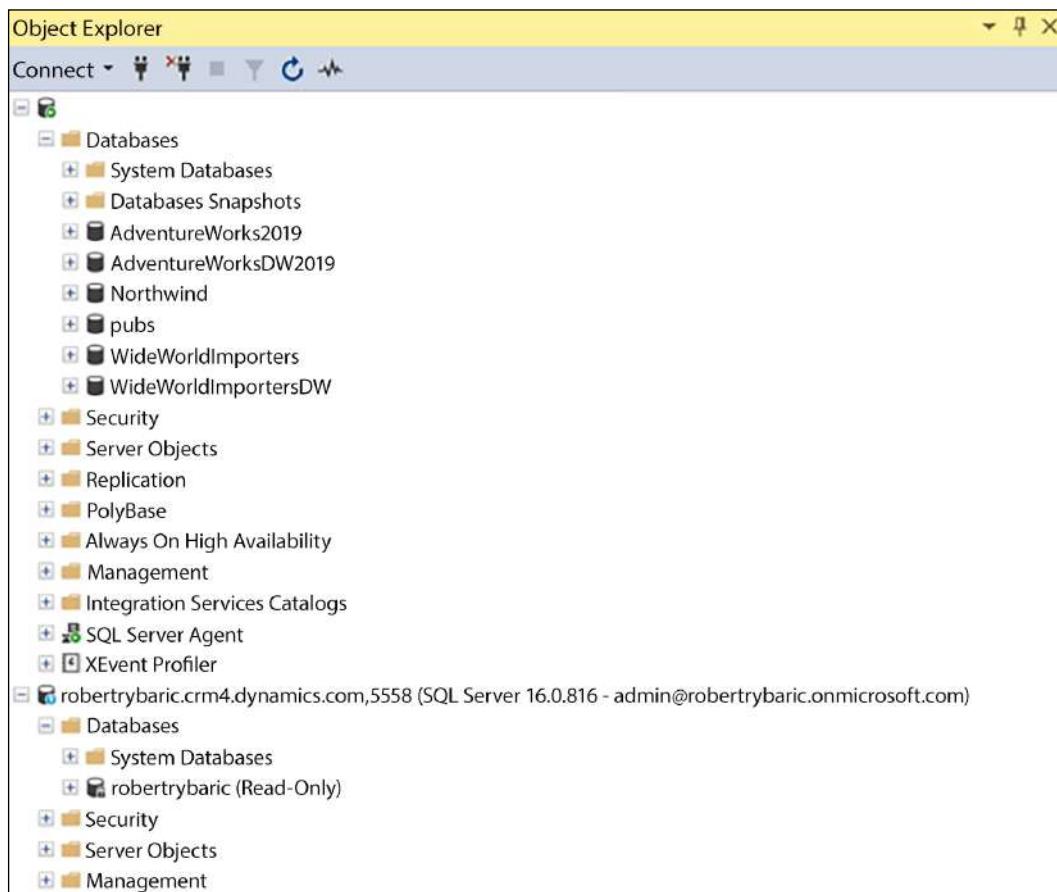


Figure 10.10: Dataverse database opened along with local databases

As we can see, the Dataverse database can now be opened with Microsoft SQL Server Management Studio in the same way as local SQL Server databases. This gives the data migration developer the possibility to easily verify all the details of each Dataverse table to be included in the data migration.

To import Dataverse data into the staging database, the *KingswaySoft* adapter can be used within an SSIS task. For this, you need to configure the connection to Dataverse and to the staging database and then create a data flow with the appropriate mapping, as illustrated in the following screenshot:

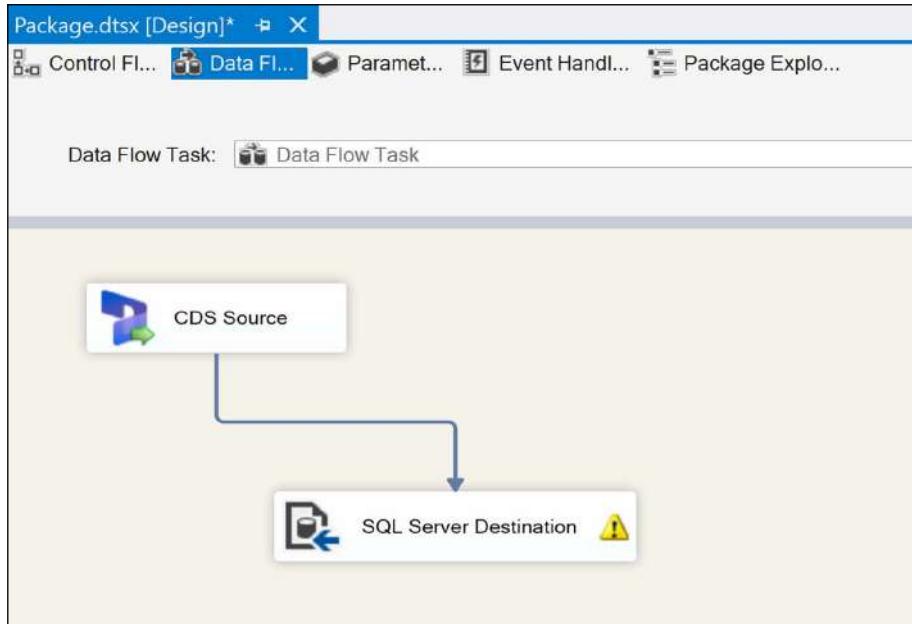


Figure 10.11: SSIS data flow

As we can see, the solution consists of a Dataverse (formerly known as CDS) data source and a SQL Server destination. The data will be loaded into the destination using proper mapping that's been configured according to the following example:

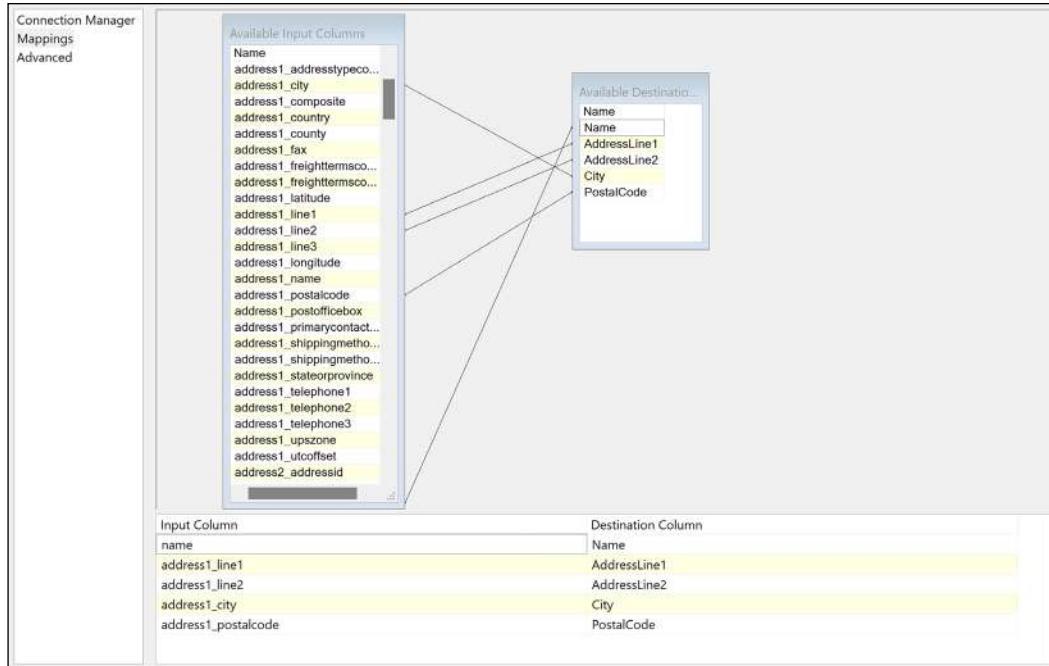


Figure 10.12: SSIS data mapping

As we can see, the tool makes it possible to perform *data mapping* between the Dataverse source and the SQL Server destination using a simple assignment tool, thus assigning the source fields to the destination fields.

These are all simplified examples that illustrate the overall approach and the capabilities of the toolset. The connection to Dataverse can either use the **SOAP** endpoint with **Office 365** authentication or the **Web API** endpoint, where registration with AAD is necessary to obtain the application ID and client secret for **OAuth** authentication.

## Transforming data

After all the external and Dataverse data is loaded into the staging database, the tricky part of the data migration process starts. We must *consolidate* and *transform* various pieces of data coming from various data sources, all of which are of a different quality, into a set of data sources so that it can be imported into Dataverse. There is no single best way to do this; expertise in both the source data structures as well as in the data transformation methods is required. There can be numerous challenges that need to be resolved in order to consolidate the data for import. You will learn about some of those challenges and best practices later in this chapter.

## Loading data

The last part of the migration process with SSIS is *loading* the consolidated data into Dataverse. For this, we can use the *KingswaySoft* adapter for the data flow destination, configured similarly to the source role. When loading data within a large data migration project, the expected number of records to load can be very high, so it is important to use all the possible options to optimize the speed of the data loading process. The *KingswaySoft* adapter supports the `ExecuteMultiple` request as well as multi-threading, as shown in the following screenshot:



Figure 10.13: Multi-threading and batch settings

As we can see, the **Batch Size** (representing the number of requests packed into a single `ExecuteMultiple` request), as well as the number of parallel threads for calling the Dataverse API, can be configured. It is important to choose proper values to avoid API limit errors by maximizing the load speed.

Since a data migration process usually requires a lot of iterations and test runs, it is required to develop all the migration steps as an automated process that can be triggered repeatedly.

In the next section, we will focus on another tool can be used for data migrations, Azure Data Factory.

## Azure Data Factory

A cloud-only tool suitable for performing large and complex data migrations is the Azure Data Factory service. This service is described as an enterprise-ready ETL and **extract-load-transform (ELT)** tool, suitable for data migration and data integration. Azure Data Factory consists of the following main components:

- **Pipelines and activities:** Pipelines consist of activities that perform a certain task, for example, to load data from a source dataset to a destination dataset, including mapping. There are many different types of actions available, like a simple data copy, invoking a data flow, an Azure Function, or a Power Query transformation.

- **Datasets:** Datasets are simply the various IT systems holding data, which need to be processed by Azure Data Factory. A dataset is always configured to connect to a defined technology such as Azure SQL, Azure Cosmos DB, Azure Data Lake, or Dataverse.
- **Dataflows:** The main job of the dataflows is to transform data in a certain way, like splitting, merging, pivoting, unpivoting, filtering, or sorting.

A very simple example of a data load from Dataverse to Azure SQL is illustrated in the following screenshot:

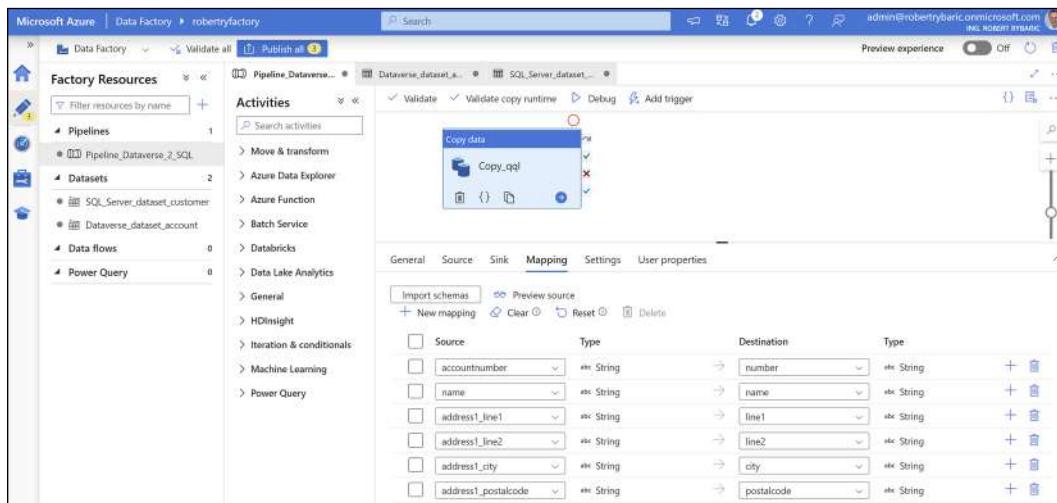


Figure 10.14: Azure Data Factory pipeline

As you can see in the preceding screenshot, a simple pipeline uses the Dataverse dataset as the source and the Azure SQL dataset as the destination. Part of the pipeline is a column mapping configuration.

With Azure Data Factory, we can build a similar solution for data migrations to Dataverse as with SSIS. The technology is, however, better suited to cloud-only migrations.

#### Important note



For further details about Azure Data Factory, please refer to the product documentation: <https://learn.microsoft.com/en-us/azure/data-factory/>.

## Data migration using code

Data migration using code is the most versatile, but also the most expensive, data migration method since significant custom development efforts would be required. This method can be used in the unlikely case where the ETL methods described previously do not offer the required capabilities. This can happen, for example, if a complex calculation that needs to be performed in real time during the migration must be implemented.

For data migration using code, the whole set of Dataverse API methods can be used, specifically the methods for inserting or updating data in the tables. The following are important considerations when using the programmatical approach:

- When records need to be updated during migration, the primary keys of the Dataverse records are usually not known in the source data. In order to reduce the number of required transactions (first, retrieving the record using a query to obtain the primary key, and then performing the update), it is recommended to implement **alternate keys**. Using alternate keys can eliminate the need to retrieve the record first before updating it.
- When the migration contains either new records or updates to existing records, it is recommended to use the **upsert transaction** type. This transaction type automatically recognizes whether to insert the record as new or update it as existing.
- Almost all Dataverse API methods are synchronous, with few exceptions, like the import of a solution or a merge of records. It is important to take the synchronous nature of the Dataverse API into consideration and plan to handle possible timeout exceptions.
- The import using the Dataverse data import wizard can be also triggered *programmatically* using the respective Dataverse API methods. Those methods are asynchronous, which avoids possible timeout issues.
- The migration code needs to cope with the **API limits** of the Dataverse API. It is therefore necessary to resolve the exceptions thrown by the limits and implement waiting conditions with retries.
- The Dataverse API supports various performance optimization features, like **multithreading** or the use of multiple transactions within one single API call using the `ExecuteMultiple` method. It is recommended to make proper use of those possibilities in the migration code.

With this, we have concluded our discussion of the data migration approaches and tooling. In the next section, we will focus on known challenges and best practices for successful data migration projects.

## Data migration challenges and best practices

Data migration is a key part of every large enterprise's Power Platform solution implementation. As part of the overall project, it is often underestimated and not planned properly. There are a lot of challenges that need to be considered and included in the project planning and execution phase, as described in the following sections.

### Planning and effort estimation

One of the most observed challenges for data migration projects within Power Platform implementations is *underestimating* the overall effort, project duration, complexity, and timing. Data migration for large enterprise solutions is always a separate project that must be carefully planned and started at the beginning of the overall project undertaking. This helps ensure the migration solution is ready at go-live time.

### Scoping the migration

With a large Power Platform implementation, in the beginning, it is often required to migrate every possible data domain into the solution so that you have Power Platform as a centralized data hub. This is, in most cases, not the correct way to understand a Power Platform solution; the solution is neither a data warehouse nor a data master for every piece of information in the organization. Therefore, it is important to carefully analyze the data architecture of an organization to understand the expectations, as well as to decide which role the Power Platform solution will play in the overall enterprise architecture of the organization. In other words, **scoping a migration** is a very important part of the planning and design phase.

In order to cover, for example, the organization's requirements for a 365-degree customer view, a lot of secondary data elements do not need to be physically imported. It is sufficient to visualize them using solution approaches such as virtual tables, embedded HTML web resources, or PCF components. The best practice to decide the scope of the data migration is to classify the data tables on whether they need to be editable, searchable, and involved in business processes in the Power Platform solution. If the answer is no, then that data table can probably be excluded from the physical migration.

Another part of scoping is to decide whether all the records from a selected data table need to be migrated or only a subset, such as those created in the last year or in the last 3 years. Older records can be considered archived and do not need to be migrated.

A good example of a bad migration design is to try to physically migrate retail transactional data into Dataverse. In industries such as retail, travel, banking, and others, every customer can generate hundreds or even thousands of individual retail transaction records. While the data itself is very interesting for a Power Platform solution, there is no reason to import that data physically into Dataverse due to the huge data volume. A much better solution would be to either visualize the data using one of the approaches mentioned previously, like virtual tables, HTML web resources, or PCF components, or to migrate aggregates from that data based on proper grouping and clustering.

## **Understanding the impact on storage**

The agreed scope for data migration has an immediate impact on the storage requirements for the Dataverse solution. As you learned in *Chapter 3, Understanding the Microsoft Power Platform Architecture*, Dataverse comes with a certain default storage capacity that depends on the number of purchased Power Platform or Dynamics 365 product licenses. It is important to perform a high-level storage requirement analysis. This helps to assess whether the available storage space will be sufficient to accommodate all the data that's planned to be imported. At this point, you may also decide to purchase storage capacity add-ons to mitigate possible storage bottlenecks.

## **Compliance considerations**

Another follow-up from scoping the migration process is to discuss **compliance**. This helps to assess whether all the data that's been planned to be migrated to Dataverse can be physically stored in a public cloud. There might be a situation where, due to some governmental or other regulations, certain data elements are not allowed to be physically stored in a cloud.

## **Understanding access issues**

Data migration often needs various data sources from legacy solutions running in the customer's own data centers or hosted by various hosting partners. For the data migration project, it is essential to get *physical access* to all systems and solutions containing relevant data sources. However, this can often be a lengthy process that requires complex approvals, security settings, access rights, and permissions to be provided, and so on. It is recommended to identify all the source systems early in the project stage to ensure access to those is granted early enough for planning, designing, developing, and testing the migration procedures.

## Coping with a lack of knowledge

Data migration must often be performed from very old legacy IT systems that are still running in production, but where *limited insider know-how* exists in the organization regarding the technical details of the solution. Since the data structures of legacy systems can be extremely complex and hard to understand, it is imperative to ensure subject matter experts are available for the project.

## Dealing with a lack of documentation

For the same reason as with the previous challenge, technical documentation for legacy IT systems is sometimes *not available* or *outdated*, thus not reflecting the current state of the solution. It is important to try to obtain as much technical documentation for the systems as possible. This helps us understand the technical details and the data model of the solution. Documenting your own data migration solution is, of course, part of the migration project.

## Poor-quality source data

It is almost a tradition for the quality of the data in the source systems to be poor and not to correspond with the high-quality data standards in Power Platform solutions. For a Power Platform implementation team, it is difficult to improve the data's quality for the migration process. The following are the typical data quality issues we have observed in data migration projects:

- Duplicate records (this issue occurs even more when consolidating data for the same data domain from different data sources)
- Mandatory fields are empty
- No unique record identifiers
- No clear relationships between data tables
- Missing record ownership information

The best practice for this situation is to make the customer contractually responsible for the data cleansing process and for the overall quality of the source data upfront. The data cleansing process can technically be performed either in the source IT systems, to make the whole migration process easier; during the migration process, if there is an opportunity to do that; or after the migration process in the Dataverse solution itself. However, experience shows us that when proper data cleansing is not performed in the source systems before the data migration happens, the customer is seldom really able to perform real post-import cleansing.

## Understanding encoding issues

Issues with encoding are rare today since most of the current IT solutions and systems out there support **Unicode data encoding**, but when planning data migration from very old legacy IT systems, this might be an additional issue. Power Platform requires all data to be encoded as *Unicode* to support the broad variety of languages being used in the world. Therefore, it might be important to perform an additional encoding step to ensure all the data is really in Unicode.

## Understanding record ownership issues

One of the specific issues when migrating data to Dataverse is the **ownership** of records. Records in every table with user/team ownership type must have an *owner*. The owner of the record is one of the most important data elements for the whole Dataverse authorization process to work properly. If the source data does not contain any useful information about the possible owner, the data must be imported in the security context of a predefined owner and the correct assignment must be performed after the migration. However, it is much better to resolve this ownership issue during the migration process. It is recommended to, when possible, include the ownership information in the source data during the transformation phase of the ETL process.

There is another specific issue that needs to be resolved if there is certain ownership information in the source data but many of the owners of the records are no longer in the organization or do not plan to use the Dataverse application. For this specific use case, **stub users** can be leveraged, as explained in *Chapter 7, Microsoft Power Platform Security*.

Another important topic to evaluate is the use of a certain user or service account, under which the migration solution should work. This account will need to be provisioned in the Dataverse solution and equipped with all the necessary access rights. This account will be the owner of every data record, for which no specific owner is explicitly defined. In this context, the *impersonation* capability of the Dataverse API can be considered.

## Understanding mapping issues

One of the main parts of the data migration process is to map the data between the source and Power Platform and specifically the Dataverse data structure. The mapping for simple data types such as **string**, **number**, **date**, and **time** is usually simple, but mapping for Dataverse-specific data types such as **choice**, **image**, **customer**, or **file** can be more challenging.

Data mapping generally consists of the following steps:

- Map the data at the **table** level (decide which source data table maps to which Dataverse table)
- Map at the **column** level (decide which columns from the source table map to which columns in Dataverse)
- Map at the **column values** (this is specifically important for columns, where the possible values are limited, such as choice, lookup, etc.)

One of the typical issues is to map the data type **choice** correctly. As we know, the choice consists of an internal numerical value and a text representation of that value. For correct mapping, the choice values must be imported from Dataverse to the staging database to be available for mapping before the main migration can be even performed.

Another important consideration is to understand that **datetime** fields are physically stored in the Dataverse database in **UTC** format. For performance optimization reasons, it is recommended to set the time zone of the user or service account performing the migration to **UTC** as well. This will avoid time zone conversions during data loading. It is also very important to understand the datetime formatting in the source IT systems, which might not necessarily be in the expected time zone. Here, a conversion would need to be performed.

## **Understanding record relationship issues**

The Dataverse database requires that we create a consistent data model including all relationships between tables. However, data in source IT systems is not always stored in database systems and respects the same level of consistency between related data tables. Simply put, parts of the records in expectedly related tables do not have the connection between them established. This issue is extremely hard to resolve and, in most cases, these records just need to be excluded from the migration process.

## **Understanding business process flow issues**

In some specific situations, you will need to assign **business process flows (BPFs)** to imported records. This can happen for records in the typical main business tables, such as *opportunities* in Dynamics 365 Sales or *cases* in Dynamics 365 Customer Service. The real challenge is not assigning the BPF itself but assigning the required stage, since the required target stage could be any of the existing stages. The typical procedure to achieve the desired BPF stage for an imported record is illustrated in the following implementation example.

The requirement is to import opportunities from a legacy CRM system into Dataverse. Each of the legacy opportunities has an equivalent of a BPF stage, where there could be stage values between 1 and 6. The implementation needs to contain the following steps:

- Import a new record into the **Opportunity** table.
- Apply the required BPF to the new record.
- Perform a forward stage change until the desired stage value is reached. (This step can be performed between 0 to 5 times over)
- Optionally close an opportunity as **Won** or **Lost**, according to the business requirements.

As you can see from this example, the whole process can consist of up to eight different transactions issued against the Dataverse API. This illustrates the complexity of certain data migration requirements.

## Understanding record status issues

Every Dataverse record has *status* and *status reason* columns by default. The *status* column defines the major behavior characteristics of the business record. Any status that corresponds to the generic status of **inactive** (it can be inactive itself or **Won/Lost** for opportunities) makes the business record read-only. This can be an issue for scenarios where it is required to import records that are designated as inactive and then make changes to the records and/or create records in the underlying tables. As an example, we could consider importing old opportunities that are mostly closed together with opportunity products. The correct sequence of import steps is as follows:

1. Import an opportunity record as open.
2. Make all the required updates to the opportunity record; for example, assign a BPF to the record.
3. Import opportunity products for the opportunity record.
4. Move the opportunity's BPF to the required stage.
5. Close the opportunity as **Won** or **Lost**, as required.

As we can see, the combination of BPF and status values can make the import process for certain tables pretty complex. This is because it consists of many individual steps in order to respect all the specifics of the Dataverse business tables.

## Setting certain system fields

Every business record in Dataverse has several system datetime fields indicating the creation date of the record, as well as the date of the last record modification. It is important to analyze the default behavior (columns will be set to the current date of data migration) and decide whether this is the desired outcome. If you need to preserve the original creation dates from the source IT systems, a specific *mapping* must be used. There is a largely unknown column in every Dataverse table that can be used instead of the standard `created_on` column. The name of this column is `overriddencreatedon`. Using this column in the migration mapping process will set the `created_on` column to the required value from the source data.

Similarly, this approach can be used to override the default value for the `created_by` field. You can do this by using the `CallerObjectId` field in the mapping.

## Migrating documents

One of the most frequent requirements for data migration is not just to migrate relational data, but also documents of any type – related or even unrelated to relational data records. Importing documents can be done either by importing them directly into the Dataverse database as annotation records with attachments or into an integrated SharePoint site collection. In both cases, the migration process requires you to identify the related business records and upload the files either to the annotation or to a SharePoint folder that is assigned to the business record. For the SharePoint-based solution, there is a need to communicate with both the Dataverse API as well as with the SharePoint API in order to achieve the desired result. For migration into SharePoint, a dedicated *SharePoint SSIS connector* from *KingswaySoft* can be used.

## Understanding the importance of the order of migration steps

A complex data migration process can include dozens or even hundreds of separate import steps into various Dataverse tables. In such a complex case, the order of the import steps is extremely important, due to the dependencies between records in relationships and other data-related considerations. The basic rule is that the import of any referenced data must be performed first. An example sequence of import steps is as follows:

1. Currency records
2. Unit records
3. Price list records (to be assigned to the parent currency)

4. Product records (to be assigned to the parent unit)
5. Price list items (to be assigned to the parent product, currency, and price list)
6. Account records (to be assigned to the parent currency)
7. Contact records (to be assigned to the parent account)
8. Opportunity records (to be assigned to the parent account and contact)
9. Opportunity product records (to be assigned to the parent opportunity and product)

The preceding list is just a simplified example. For large migration projects, the dependency list can be much larger, also containing a lot of custom tables. As shown in the previous example, there is already a hidden relationship issue when we need to set the primary contact for every account. This is not possible only following record creation steps, since the contacts do not exist during account creation. A record update step would need to be inserted into the sequence according to the following example:

1. Account records (to be assigned to the parent currency)
2. Contact records (to be assigned to the parent account)
3. Updating the primary contact of accounts
4. Opportunity records (to be assigned to the parent account and contact)

The preceding example illustrates the importance of using the correct sequence of data migration steps in order to achieve the desired results.

## **Understanding migration automation**

Since complex data migration usually consists of a multitude of steps – even hundreds of individual steps – the whole migration process must be automated. Every complex migration must be developed and thoroughly tested for the final migration run in the production environment. The number of individual test migration runs can be also considerably high in order to correct every identified mistake in the *migration logic* or *technological* approach. To manage this complexity, you must fully automate the whole migration process to make it easily repeatable and reliable. Automating a complex migration can be achieved using several different approaches, such as using the *SSIS data integration project*, using *PowerShell*, or using *custom development* to call all the required actions with code.

## Understanding migration performance

As you learned in the previous chapters of this book, Dataverse offers certain API endpoints to communicate with. These same endpoints must also be used for data migration purposes since no direct writing access to the underlying Azure SQL database is available. That being said, it is very important to consider the overall data migration duration when planning the final migration before going live. The Dataverse endpoints, Web API, and SOAP provide certain transaction performance that is also limited by the platform API limits, as described in the next section. To achieve the best possible migration performance, all these technical limitations must be carefully considered. To maximize the data load performance, the options for using the `ExecuteMultiple` request type and multi-threading need to be used and configured. The best-performing combination of the mentioned options needs to be tested. Finally, to evaluate the final migration performance, full test execution is necessary.

### Important note



For further details about using the `ExecuteMultiple` method in code, please refer to the product documentation: <https://learn.microsoft.com/en-us/power-apps/developer/data-platform/org-service/execute-multiple-requests>.

A specific topic to consider when planning a data migration and preparing the design is to decide which types of automation can be, or must be, disabled during the import's execution. A complex Dataverse solution can have various types of automation implemented (workflows, plug-ins, auditing, duplicate detection rules, Power Automate flows, and others). Some of this automations must be switched off due to business requirements since they should only be executed in normal operation. Some of the automation must be switched off due to the negative performance impacts they might have on record creation and updating transactions.

## Resolving API limits

As you learned in *Chapter 3, Understanding the Microsoft Power Platform Architecture*, the Dataverse API has some protective limits regarding how many requests a single process can trigger against the API within a defined time period. This is particularly unfortunate for large data migration projects that need to create an expectedly high number of records.

**Important note**

For further details about handling the API limits within a data migration solution, please refer to the product documentation: <https://learn.microsoft.com/en-us/power-apps/developer/data-platform/api-limits>.

One possible option to overcome this limitation is to request that these limitations be lifted for the period of data migration. This can be done using the *Microsoft customer support* channel.

## Time for migration execution

For large migration projects, the time required to execute the full migration can be significantly longer, taking the usual speed of the Dataverse API into consideration, combined with API limits, as described in the previous sections. Therefore, it is very important to test the real duration of the migration, use all possible technical options to increase the migration speed, and set the right expectation for the customer. Large data migrations are usually planned to be performed outside of business hours, on weekends, or during a dedicated migration time. It is very important to ensure that, before starting the final data migration process, all the existing IT systems are stopped so that no new data is generated during the migration process.

If one-time migration is impossible to organize due to operational reasons – certain IT systems cannot be stopped for a longer period of time (several days), for example – it might be useful to think about a multiple-step migration, where the data is migrated in several phases and each phase is therefore shorter in terms of execution time. This approach is very challenging and requires very detailed planning so that you can decide which data in which sequence can be imported in every migration phase.

## Verifying the data by the customer

The last but most important step in the data migration process is to verify the data quality of all migrated data. The final verification process and sign-off must happen before the final migration into production is performed and must include all the required production data. This step is the full responsibility of the customer. The implementation partner must require this sign-off as a prerequisite to trigger the **final migration process**.

With that, you have learned that a complex data migration project can have a lot of challenges and that it needs to be planned and staffed properly. You have also learned about several best practices, as well as how to overcome typical challenges.

Finally, for the last time, we will have a look at our fictitious customer Contoso Inc. to see how they have adopted the information from this chapter, as well as how their data migration design will look.

## Contoso Inc. data migration design

The project team for Contoso Inc. has carefully analyzed the possibilities for data migration and the recommended best practices. They now fully understand that data migration is a complex undertaking and needs proper planning, resourcing, and an appropriate timeframe. They know that the first step is to perform an analysis of their IT systems and solutions inventory to help them figure out where the important data is, and how it can be incorporated into the data migration project.

After they performed this analysis, it turned out that their data landscape was very diverse and complex, and that the data migration process would not be simple. The following were their key findings:

- Within the whole organization, there are approximately 20 global and regional IT systems holding relevant customer data. The technology of these systems is very different, most of this data is, however, stored in various relational databases, such as Microsoft SQL Server, Oracle, MySQL, and IBM DB2. A small number of customer databases are stored in legacy IT systems, though there is the possibility to export that data as TXT files.
- Their existing IT systems do not hold any contact data that's worth importing. The contact records are rarely used, and their overall quality is poor. However, there is a good database of contact records in the Exchange address book. Contoso Inc. decided to use the **Dataverse server-side synchronization** capability to bring the contacts into Dataverse. They will train the end users to synchronize their most important and high-quality contacts into Dataverse using the **Dynamics 365 App for Outlook**.
- The product catalog is part of 10 various global or local IT systems. There will be a need to consolidate this data into the Dataverse product catalog.
- Contoso Inc. and its regional subsidiaries use more than 10 various sales management solutions to track leads and opportunities. This data will need to be consolidated and migrated into Dataverse. They've decided to import leads, opportunities, and product line items from the last 3 years since within this time period, there are still some open opportunities that need to be followed. In most of the legacy sales solutions, there is some form of sales stage information. It will be necessary to consolidate the sales stages and use them for the imported leads and opportunities.

- They are also using around five legacy customer support IT solutions that contain very valuable data about active and closed service cases. There are also many data elements that incorporate knowledge articles. This data must also be consolidated and imported into the Dataverse solution. Since Contoso Inc. is eagerly waiting for the possibility to analyze the efficiency of its overall customer support, they have decided to consolidate and import all their existing support cases from the last 5 years. The knowledge base-like data will be consolidated and imported into Dataverse as well.
- Contoso Inc. does not seem to use a professional marketing solution in their organization. In most cases, they are using proprietary mass mailing solutions, where there is little to no relevant data to be migrated.
- Since they also have an important business area in field service, they have analyzed the tools that are being used in the organization and found only proprietary tools such as Excel or various calendar apps, which are used for planning field service activities.
- Contoso Inc. relies heavily on using documents in the context of its legacy business applications. These documents are stored in various local or centralized repositories, and in most cases, there is a reference between the document and a business record in a database. They need to consolidate all those documents and upload them onto the Power Platform solution, in an integrated SharePoint repository.

Besides the most important data tables that were identified during the analysis, they documented a large number of various additional business and configuration data. This was stored, to some extent, in the existing legacy applications, but also in Excel files or other local repositories. This data needs to be consolidated and, after further decision-making, imported into Dataverse.

The analysis clearly indicated that the data migration process would be complex and needs to use proper tooling. That's why they decided to use the recommended approach of *SSIS* and all the necessary connectors from the ISV partner *KingswaySoft*. The required product licenses will be purchased, and the data migration project team will undertake training to use those tools in the best possible way.

The whole data migration setup will be configured according to the following diagram:

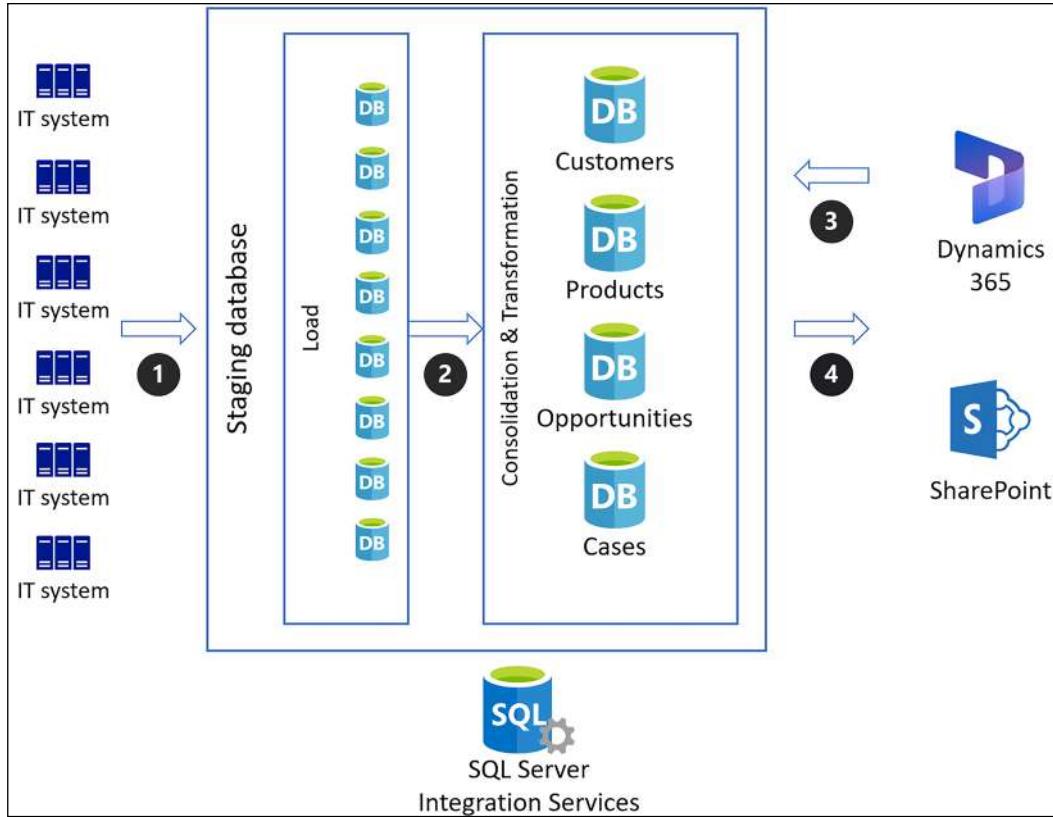


Figure 10.15: Contoso Inc. data migration setup

As we can see, the project will use the SSIS-based setup and will consist of the following major steps:

1. The various data types will be imported into the staging database to a number of specific intermediate database tables. They will use the SQL Server capability of linked servers for every data source, which is stored in a relational database and imports the data using SSIS. For legacy data sources, they will use the text files that have been exported from the applications and import them using SSIS.
2. Any data that's of the same type will be consolidated into the target importable tables. Since the most important data tables are currently stored in multiple IT systems, the database tables they created in the previous steps must be imported and consolidated into the target database tables using SSIS data transformation tasks. This helps achieve a unified data structure.

3. Any data required from the Dynamics 365 database will be imported into the staging database and consolidated in the importable tables. They will analyze the need to have Dataverse foreign keys in the target importable database tables. For every requirement of this kind, the content of a Dataverse table will be imported into the staging database using SSIS with the *KingswaySoft* adapter. After the import, the foreign keys will be restored in the target importable tables using SQL commands or SSIS.
4. Finally, the consolidated data will be imported into the Dynamics 365 database. After the content of the target importable tables in the staging database is completely ready for import into Dataverse, they will perform the import jobs using SSIS with the *KingswaySoft* adapter.

Besides relational data migration, the required document files will be prepared and relationships to the target importable tables will be recreated. After the relational data has been imported into the Dynamics 365 database, the files will be imported into the integrated SharePoint site and references to the Dynamics 365 records will be created. The import will be performed using SSIS with the respective SharePoint adapter from KingswaySoft.

Regarding timing, Contoso Inc. has analyzed both single-step as well as multiple-step migration. They understand that a single-step migration will be very challenging, but they do not want to spend additional time and effort dividing the migration into multiple steps. Proseware Inc. experts were asked to perform a proof of concept to evaluate the possible maximal migration durations. Based on their results, they decided to migrate in a single step and to plan for a 2-week freeze of all the IT systems included for the migration period.

After this last phase of the overall solution design, Contoso Inc. is now confident that they have covered all the important aspects of the Power Platform solution. With all this knowledge and architecture and design documents, they are now ready to take the challenge and start the execution of the project's implementation.

## Summary

In this last chapter of this book, you learned a lot of useful information about **migrating data** into a Power Platform solution. Data migration for small and simple projects can be easy and requires minimal effort. However, migrating data from a wide variety of sources into a complex enterprise and global solution is a serious undertaking that requires enough time, planning efforts, and skilled resources. For this reason, you learned about the different tools you can use for migrating data, as well as their usage scenarios.

In the last part of this chapter, you came across a lot of typical challenges and ways to mitigate them in order to be successful and deliver a smooth data migration project.

With this knowledge, you should be well prepared to master very complex data migrations and successfully complete a large Power Platform project.

*I wish you all the best on your Power Platform journey!*

## **Share your thoughts**

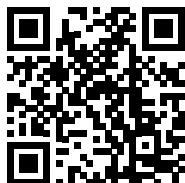
If this book is helping you improve your skills, we'd strongly suggest leaving a review on [Amazon.com](https://www.amazon.com). This helps us know if you like our work and if the chapter content has been valued, and also helps the buyers on Amazon know if the book is right for them.

So, everyone else benefits from your review - we wouldn't want you to miss out. You can now reach out to [review@packt.com](mailto:review@packt.com) with a screenshot of your review and the book URL, and we'll send you a \$5 voucher for your next Packt purchase. Thank you in advance for engaging with us, we are excited to see your review!

## **Learn more on Discord**

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/businesscenter>



---

# Appendix

---



# Best Practices for Solution Architecture

The following sections are an arrangement of all the best practices we have provided for the key topics found in this book. These practices have been collected for anyone who needs advice concerning their Power Platform solution but does not need to revisit the groundwork knowledge in the relevant chapters. Although these practices are the result of decades of experience, strive to be both practical and fact-driven, and are generally accepted, it should be acknowledged that there is some room for subjectivity on all these fronts and some situations may call for a different approach.

Below, we will cover the best practices for all major areas of Power Platform solution architecture, including the following:

- Architectural best practices
- Application lifecycle management best practices
- Security best practices
- Extensibility best practices
- Integration best practices
- Data migration best practices

We will begin with advice surrounding the architecture of Power Platform solutions, since it is important to have the Microsoft services, tenants, and environments configured before you can understand what considerations will have to be followed for – for example – security and extensibility.

## Important note



These practices can also be found in the relevant chapters highlighted throughout. The purpose of this appendix is to provide you with the additional option of viewing these practices in one place.

## Architectural best practices

In *Chapter 3, Understanding the Microsoft Power Platform Architecture*, we discuss the architecture, clients, and administration and monitoring that are necessary for setting up the Power Platform ecosystem. The best practices in this chapter have been consolidated below.

### When to use multiple tenants

An Azure Active Directory tenant is the highest node in the customer's cloud ecosystem. Usually, when a new customer purchases a Microsoft cloud service, they obtain their new tenant as part of their subscription. When purchasing additional Microsoft cloud services, they are added to the tenant, which was created with the first purchase. Although the best practice is certainly to *use a single tenant wherever possible*, there are situations where a single tenant is not sufficient for a customer, and they need additional tenants. This is specifically the case when they plan to implement certain on-premises active directory integrations with the cloud. The following could be possible reasons for additional tenants:

- Separate development and testing Azure Active Directory environments are required.
- The customer has a complex enterprise ecosystem with a very heterogeneous Active Directory structure.
- The customer requires Active Directory integration using a topology that is not supported for single-tenant integration.

In the following sections, we will explain these scenarios for using multi-tenant environments a little further.

### Separate development and testing environments

Large customers might require a permanent development and/or testing environment for Active Directory, specifically to develop, test, and maintain the Active Directory integrations with the cloud. They might want to develop and test specific security and data protection solutions before they go into production on the main production tenant. In those situations, it is possible for a single customer to purchase multiple Azure Active Directory tenants from Microsoft using special contractual agreements.

Technically, this would be a set of independent integrations between multiple on-premises Active Directory instances and multiple Azure Active Directory tenants, with a separate set of user accounts in every such integration, as shown in the following diagram:

Microsoft cloud	 Azure Active Directory	 Azure Active Directory	 Azure Active Directory
Customer on-premises infrastructure	 ADFS Proxy  ADFS  Active Directory	 ADFS Proxy  ADFS  Active Directory	 ADFS Proxy  ADFS  Active Directory
	Development	Testing	Production

Figure A.1: Multi-tenant environment

When using this multi-tenant approach, any cloud services such as Microsoft Power Platform are usually deployed **only in the production tenant**. If so, the above setup is free from any typical multi-tenant issues like multiple not-synchronized user accounts, inability to perform centralized cross-tenant administration, and so on.

## Unsupported integration topology

Another more complex situation occurs when, for whatever reason, the complexity of the customer's active directory environment is not supported for integrating with a single Azure Active Directory tenant. Azure Active Directory supports many different topologies, as described in the following product documentation:

<https://docs.microsoft.com/en-us/azure/active-directory/hybrid/plan-connect-topologies>.

However, certain topologies are not supported, specifically the following:

- Using several Active Directory synchronization services against one single Azure Active Directory domain

- Any configuration where the same active directory object (user account, and so on) would be synchronized into multiple Azure Active Directory tenants
- Any cross-domain synchronizations between multiple active directory forests and Azure Active Directory tenants

These complex situations might dictate the use of multiple tenants for **production use**. Distributing any cloud solutions – and Power Platform applications are no exception – across multiple tenants for production purposes imposes some serious issues that need to be taken into account:

- Within the same tenant, accessing different Power Platform environments with the same users is easy and is just a matter of security settings. However, Power Platform environments in separate tenants are totally separated and cannot be accessed with the same user credentials. A user would need separate and non-integrated credentials to be created in every tenant.
- While, within the same tenant, certain administration tasks are easy to perform (for example, copying one environment into another), for environments in separate tenants, this is not possible.
- Data integration for environments in the same tenant is much easier to achieve than it is for environments in separate tenants, simply due to the need for managing record ownership over separated and non-integrable user groups.

As said before, the best practice is to *avoid using more than one tenant* when establishing Power Platform solutions in an organization. If there is a need to have multiple Power Platform environments for production purposes, the preferred way is to have them all in the same tenant. Based on your requirements, the environments could be provisioned in the appropriate cloud regions for better performance, data residency, and other reasons.

If that still does not work and a multi-tenant topology is inevitable, then for Power Platform solutions, the following options are possible:

- Central consolidation environment
- Central reporting environment

In the following sections, we will describe the possibilities of these two options.

## Central consolidation environment

For this option, a central consolidation Power Platform environment would need to be created, along with custom data integration solutions, in order to consolidate all or selected data in the central environment, as illustrated in the following diagram:

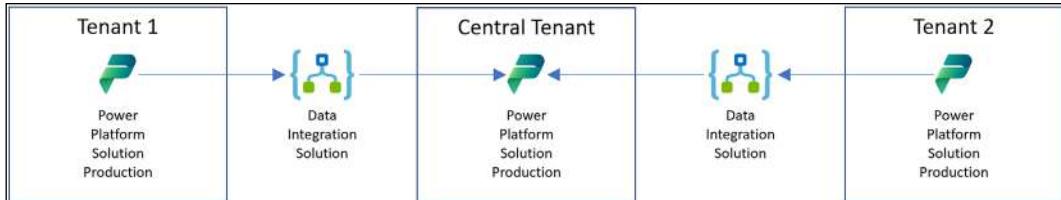


Figure A.2: Central consolidation environment

The example approach shown in the preceding diagram consists of three tenants and in every tenant, there is a Power Platform environment. In order to achieve a data consolidation in the central tenant, we need a custom data integration solution from the two satellite tenants into the central tenant.

This solution will work under the following circumstances:

- The user groups in all the tenants are *separated and non-integrable*, where the satellite tenants might represent regional subsidiaries and the central tenant user group represents the headquarter management and central services.
- The data integration solutions consolidate the data from the tenant-based solutions into the central solution, mainly for *read-only* purposes. Any data modifications in the central tenant would not be replicated in the solutions in the satellite tenants.
- The data integration solutions would need to perform an appropriate *user mapping*, since the user groups are separated and non-integrable and there is a need to establish ownership for every record in the Power Platform solution in the central tenant.

A more complex version of this approach would be if the requirements were to dictate a bi-directional data integration, but fortunately, this is not typical.

## Central reporting environment

Another consolidation solution would need to implement a centralized reporting component based on Power BI, along with custom data integration solutions, to consolidate all or selected data in Power BI, as illustrated in the following diagram:

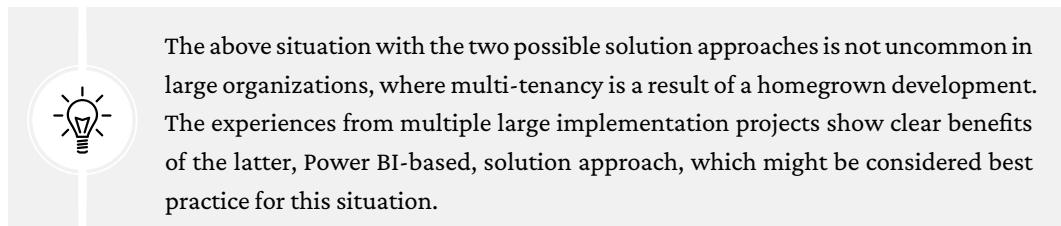


Figure A.3: Central reporting environment

The example approach shown in the preceding diagram consists of three tenants. In the two satellite tenants, there are Power Platform environments, while in the central tenant, there's a Power BI solution. In order to achieve data consolidation in the central tenant, a custom data integration solution is needed from the two satellite tenants into the central tenant.

This solution will work under the following circumstances:

- The user groups in all the tenants are *separated and non-integrable*, where the satellite tenants might represent regional subsidiaries and the central tenant user group represents the headquarter management and central services.
- The data integration solutions *do not need to perform user mapping*, since Power BI has a different security concept, and keeping the original record ownership information can be beneficial for analytical and reporting purposes.



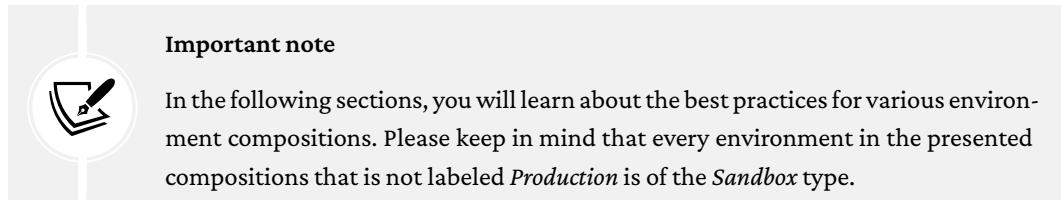
The above situation with the two possible solution approaches is not uncommon in large organizations, where multi-tenancy is a result of a homegrown development. The experiences from multiple large implementation projects show clear benefits of the latter, Power BI-based, solution approach, which might be considered best practice for this situation.

## Environment strategies for an enterprise-scale project

Designing an environment strategy for complex and long Power Platform implementation projects is not trivial and requires a lot of parameters to be considered. In this section, we will present the best practices for setting up an environment ecosystem for developing and operating Power Platform solutions.

In *Chapter 3, Understanding the Microsoft Power Platform Architecture*, we provided an overview of all possible Power Platform environment types. But for enterprise projects, only the following two are real candidates to be used when setting up an environment strategy:

- Sandbox
- Production



### Important note

In the following sections, you will learn about the best practices for various environment compositions. Please keep in mind that every environment in the presented compositions that is not labeled *Production* is of the *Sandbox* type.

These best practices are generally valid for any Power Platform solution, but specifically for Dataverse-based solutions, due to the additional complexity Dataverse is bringing into the equation.

## The shared test and production environment strategy

This environment strategy can be used in smaller and less business-critical solutions, where sharing with other solutions does not present any risk. The environment setup is presented in the following diagram:

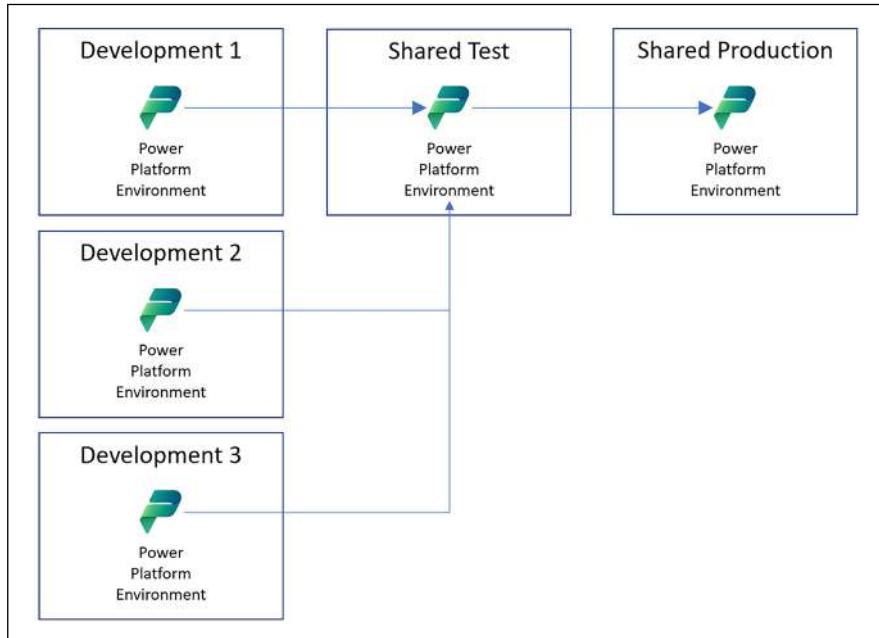


Figure A.4: Environment setup for shared test and production

As you can see in the diagram, there are multiple independent development environments, where various teams develop smaller solutions. Once the solutions are ready to be tested, they move to the shared test environment. Later, when finished, they move to the shared production environment, where they operate together with the other parallel solutions. This strategy requires proper administration, security setup, and governance, specifically around the security setup in Dataverse and regarding the proper DLP policies specified to ensure a balance between security requirements and usability.

## The dedicated environments strategy

Dedicated environments are usually used for complex, business-critical solutions. The word *dedicated* means that there is an environment setup exclusively for the single solution. The simplest dedicated environment setup is presented in the following diagram:

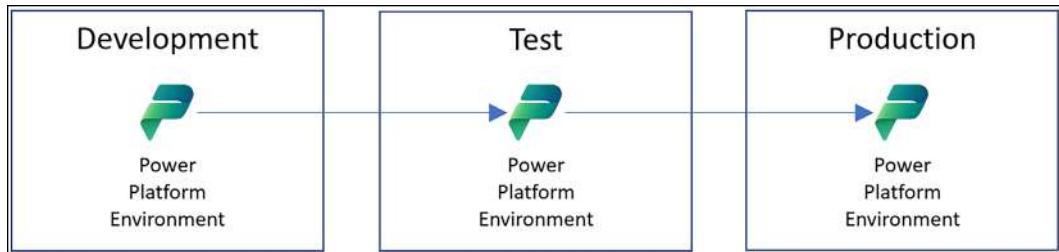


Figure A.5: Environment setup for dedicated environments

As you can see in the diagram, the simplest setup contains the development, test, and production environments, all used for the single business-critical solution.

These environments are used for the following purposes:

- **Development:** This environment is used by the development team to develop the solution, including unit testing.
- **Test:** This environment is used for all testing except unit testing, such as system integration testing or user acceptance testing. In the case of any integrations, this environment is integrated with the customer's test versions of their IT solutions.
- **Production:** This environment is used for production purposes and is integrated with the production versions of the customer's IT systems and solutions.

A dedicated environment setup can be much more complex than the presented one, depending on the complexity of the solution, the development teams, the testing requirements, and more. Let us analyze the different factors mentioned and present some best practices.

## Complex solution/complex team structures

If the developed solution is very complex, or the team structure is very diverse, it might be necessary to have more than one development environment in the setup. Examples of these situations can be:

- The solution requires significant custom development, where the developers need dedicated development environments to be able to develop and at the same time not disturb or be disturbed by the ongoing standard customizations.

- The complex solution is broken down into sub-projects consisting of separate, independently-developed workloads, for example, basic customer management, sales management, customer service management, etc. In this situation, multiple development environments can also be necessary.
- There are distinct teams developing the solution, possibly from different companies, again requiring separate environments.

In all these situations, it is necessary to be able to manage the solutions coming from the various environments and consolidate them into a consistent solution package being able to be transferred to the downstream environments. This can be achieved either by using a development master or by properly handling the solutions in a source control system, like Azure DevOps.

## The complex testing strategy

For customers that require several stages of testing, an appropriate testing environments cascade can be implemented:

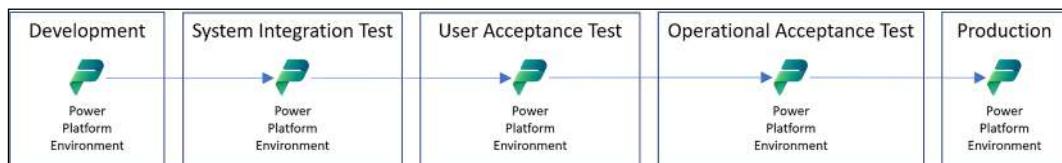


Figure A.6: Environment setup for complex testing scenarios

In the preceding diagram, there is a three-stage testing cascade of **system integration testing (SIT)**, **user acceptance testing (UAT)**, and **operational acceptance testing (OAT)**, which is a typical required setup for some large customers. The purposes of these environments are as follows:

- **SIT:** This environment is used for technical testing of the solution as a whole, including all possible integrations with other IT systems.
- **UAT:** This environment is used for final end-to-end functional testing provided by human testers.
- **OAT:** This type of testing is used to verify the operational readiness of the solution to be supported so it can become part of the production environment.

## The multiple release strategy

In the case of a large and complex Power Platform solution, where multiples solution releases are planned, it is necessary to ensure that there are two development and testing streams:

- **Main development stream:** For developing the next major solution release (version “ $N+1$ ”)
- **Support development stream:** For bug fixing and minor improvement purposes for the existing solution’s release in production (version “ $N$ ”)

This setup can be seen in the following example diagram:

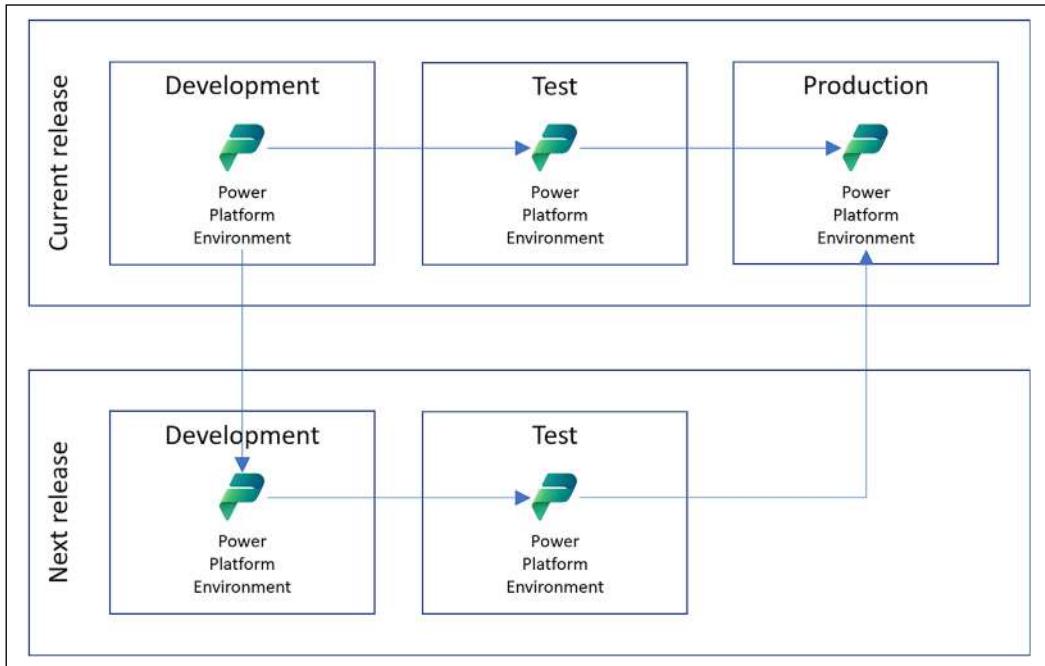


Figure A.7: Environment setup for the multiple release strategy

While, in the *Current release*, the development and test environments are used to resolve any possible issues and bug fixes, in the *Next release* the development and test environments are used to develop the next solution version that will be released in the future.

There must be a structured way to properly transport all the identified bugs and other issues from the support stream into the main development stream. This helps to ensure that those issues will not be replicated in the next major solution release. There is no best way to do this; it is usually managed using a bug-tracking system.

## The product upgrades strategy

In the case of a longer running solution implementation, where a new Power Platform product release is expected during the implementation, it is necessary to ensure that a separate environment for development and testing is established. This helps verify the full compatibility of the developed solution with the new Power Platform product release. For those dedicated environments, the preview feature must be activated at the beginning of the preview window (between the preview's availability and the new product's release date) in order to ensure enough time for all necessary tests.

This setup is shown in the following example diagram:

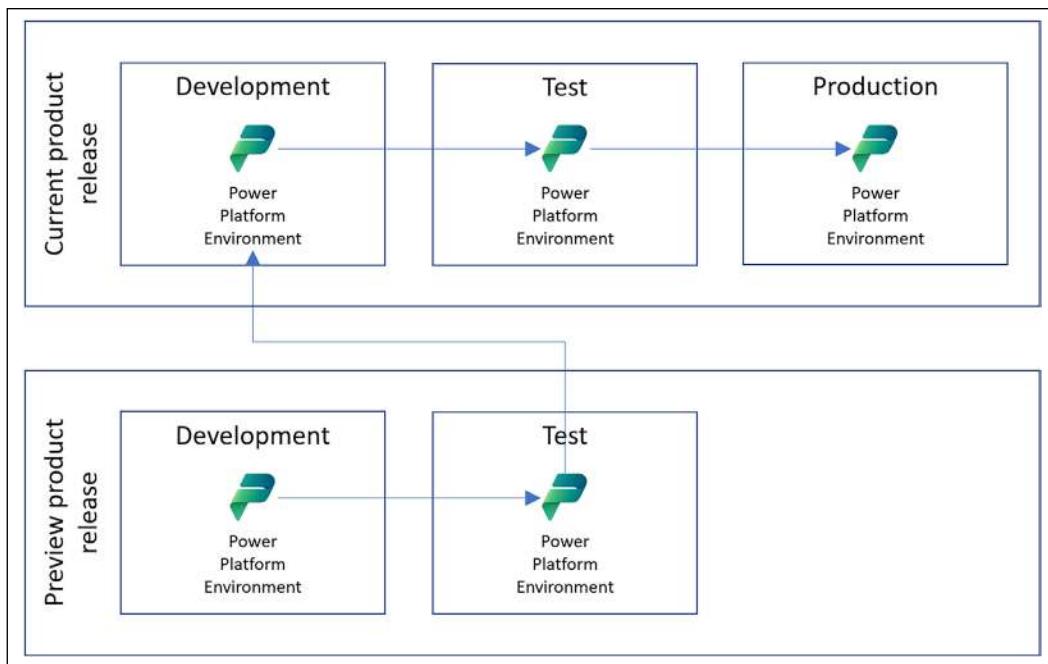


Figure A.8: Environment setup for product upgrades testing

The *Current product release* is used to develop the solution on the Power Platform's major product release, also called **general availability (GA)**. The *Preview product release* must have the Power Platform product preview feature activated. It is used to test the current solution on the potential compatibility issues of the solution with the next Power Platform product release.

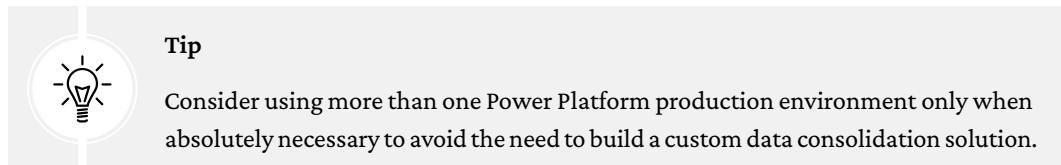
For this setup, there must be also a way to properly transport all the identified bugs and other issues from the product preview environment into the main development environment. A proper bug-tracking solution can be used to track these findings and provide input to the development team.

## Environment regions

As you learned in *Chapter 3*, every environment can be created in a selected Power Platform region, independently of the region of the tenant. This is a very useful feature, making it possible to fulfill specific requirements:

- **Data sovereignty requirements:** In certain countries, there might be a legal requirement to store data of some types only within the borders of that country.
- **Performance requirements:** The distance between the users and the location of the Microsoft data centers influences the overall performance of the Power Platform solution. This is due to the latency time between the two points. In order to optimize the performance, it is useful to select a Power Platform region near to the majority of the users of the solution.

While it is possible to specify the best regional environments configuration using multiple production environments in various regions, it is also necessary to keep the possible **data consolidation consequences** in mind, since there is no out-of-the-box solution for automated data synchronization between Power Platform environments.



### Tip

Consider using more than one Power Platform production environment only when absolutely necessary to avoid the need to build a custom data consolidation solution.

## Administration and monitoring

Power Platform is being built more and more for **citizen developers** and **power users**. The goal is to break the barriers of large and long-running IT projects and empower users to bring business value fast by letting them develop small solutions in a low-code/no-code fashion. This is certainly a great idea, but there are also some risks with the uncontrolled distribution of app maker rights into an organization. We need to consider specifically the following risks:

- Governance over environment creation
- Security risks connected with *shadow IT* (uncontrolled creation of IT components by non-IT personnel)
- Data protection risks regarding the unrestricted use of connectors
- Maintenance of the apps and flows created, and more

There are various approaches, tools, and methods we can use to mitigate these risks, as described in *Chapter 3*, such as the proper use of DLP policies or using monitoring tools. To support customers so that they can achieve proper governance, Microsoft offers a very useful **Center of Excellence Starter Kit** for Power Platform: <https://aka.ms/COESTarterKit>.

The kit consists of a collection of Dataverse solutions, model-driven apps, canvas apps, Power Automate flows, connectors, Power BI report, templates, and so on. These are designed to help customers with large and growing Power Platform ecosystems in the following areas:

- Managing an inventory of all Power Platform tenant resources
- Managing DLP policies
- Managing the auditing of apps or flows
- Analyzing connector usage
- Managing unused apps
- Onboarding app makers

It is highly recommended to adopt the starter kit since it significantly accelerates the process of establishing overall Power Platform governance in a complex ecosystem.

The following screenshot shows an example of one of the many Power BI dashboards contained in the CoE Starter Kit, which is providing an environment overview:

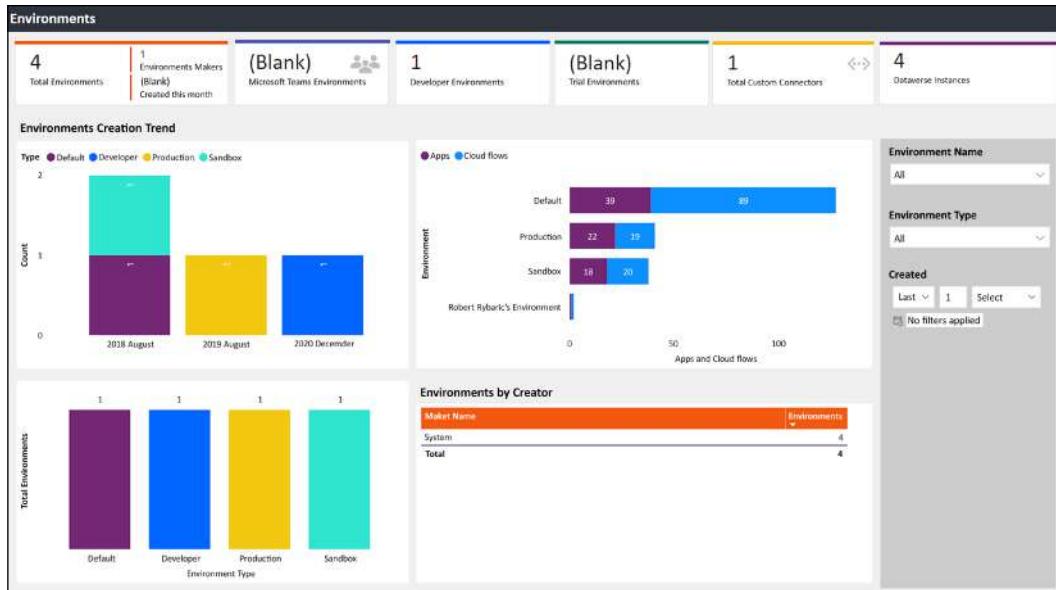


Figure A.9: CoE Starter Kit – Environments overview

The dashboard presented in the preceding screenshot is providing a detailed overview of environment creations by type on a timeline, the number of apps and flows per environment, most active app creators, and more. The preceding example illustrates one small part of the offering, but the whole **CoE Starter Kit** is much more comprehensive and provides real help for organizations planning large Power Platform implementations.

#### Important note



To specifically support the automation governance when using Power Automate, Microsoft is bringing a new starter kit called Microsoft Power Platform Automation CoE Starter Kit. As of writing this book, the starter kit is in preview. The kit has a similar structure and purpose as the Power Platform CoE Starter Kit, but strongly focuses on all aspects of the Power Automate product.

## Application lifecycle management best practices

In *Chapter 5, Application Lifecycle Management*, we learned about the two key ALM pillars for Power Platform solution development: *solution management* and *ALM automation*. In this section, we have consolidated the best practices for these topics.

## Create a specific solution package

It is the best practice to always create a specific solution package for a defined purpose and not use the **default solution** or just try to customize the environment outside of any dedicated solution package. While it is possible to include existing components in a solution package at any time, this should be only used to include standard components and never for custom components. Custom components should be always created in the context of a solution from the beginning. Strictly following this best practice will avoid the misconfiguration of components, for example, using the wrong prefix for custom artifacts or forgetting to include certain artifacts in a solution.

## When to use unmanaged or managed solutions

There is a clear purpose and very strong recommendations from Microsoft regarding the use of the proper solution type:

- **Unmanaged solutions:** These should be used **only** within the development environments, regardless of how many there are and the complexity of their structure.
- **Managed solutions:** These should always be used when the solution leaves the realm of the development environments and starts the journey to the downstream environments (various test environments, production, and so on).

The use of the proper solution type is one of the most controversial best practices, which is still not always respected, mainly for historical reasons, when the managed solutions approach was not mature enough and caused increased complexity and various issues.

## When to use a single solution

Using a **single solution** package for the whole Power Platform solution is a viable way for most of the situations where there are no dependencies on existing solutions or when the overall solution is not extremely large and the manipulation (export and import) with the solution package would not take an unreasonably long time.

## When to use multiple solutions

Using **multiple solutions** always imposes a risk of creating collisions and dependencies and making the solution imports, exports, and deletions impossible. The only justification for using multiple solutions is when those solutions would be able to 100% avoid unwanted mutual dependencies.

Multiple solutions can be implemented in different typical situations, for example, when developing a complex Power Platform solution covering several workloads, and there is a need for specialist teams to develop the respective workloads independently from each other, as illustrated in the following diagram. But in such a case, it is also highly recommended to strictly define the customization boundaries for each specialist team to avoid conflicts:

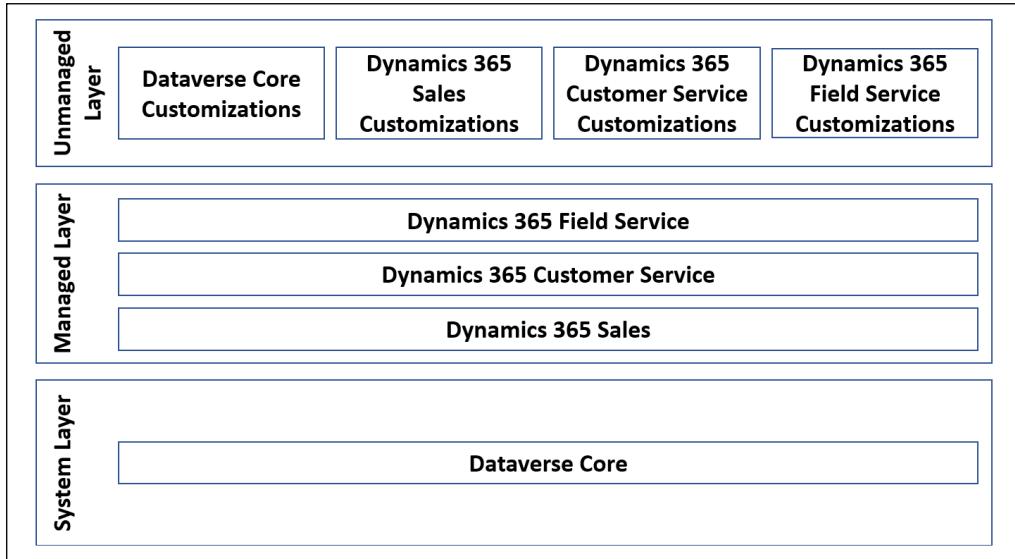


Figure A.10: Workload-based multiple solutions approach

Another example of implementing multiple solutions could be for a **global solution** deployed into separate regional environments with certain per-region deviations in the user interface, business processes, and so on. This use case will intentionally create dependencies, but in this case, those dependencies are known and must be managed properly. An example of such an approach is illustrated in the following diagram:

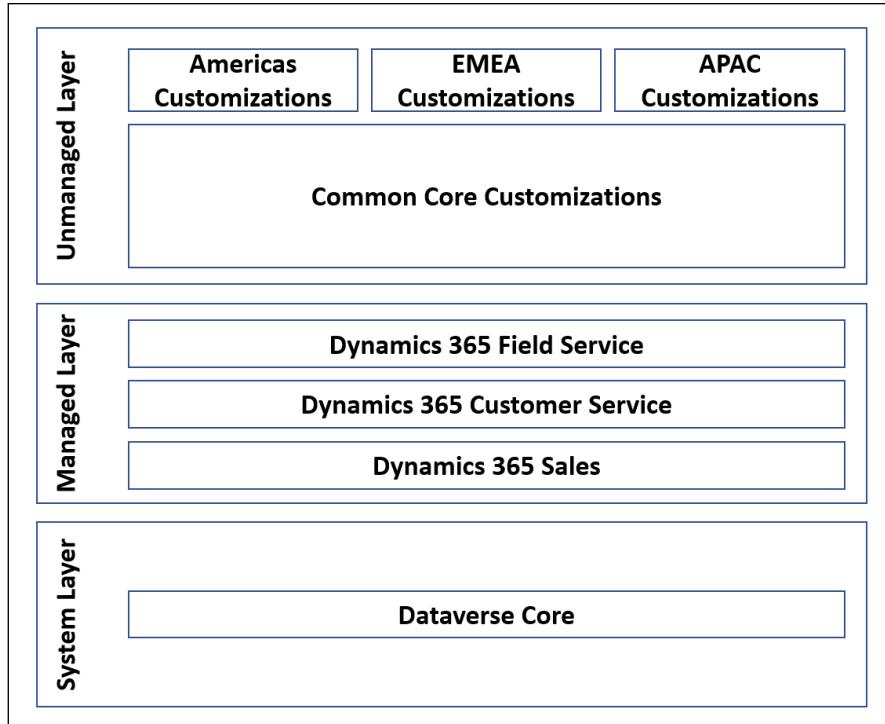


Figure A.11: Regional-based multiple solutions approach

When using this approach, it is recommended to follow some best practices:

- Keep the custom functionality as much as possible in the **Common Core Customization**.
- Keep the **regional customizations** as small as possible.
- Do not implement any data model changes in the **regional customizations**, only additional UI elements and business processes.
- The regional customizations may have dependencies on the Common Core Customization but never in the other direction.
- Avoid any mutual dependencies between the regional customizations.
- Use solution segmentation to the highest possible extent to avoid collisions.

## Component sharing and component libraries

In the case that the implementation organization follows a strategy of creating libraries of **reusable components**, then it makes sense to place the reusable components into a separate solution package. These solution packages would usually have their own independent ALM. The whole management of these packages would follow product development best practices rather than a typical project implementation lifecycle. Such solution packages would need to be based on standard solution components only and be strictly independent of any custom artifacts in other solutions. To manage dependencies properly, such solution packages would need to be imported first to every environment where they would be used.

## Using segmentation

Segmentation can be used for Dataverse tables, as complex components containing subcomponents. When using segmentation, there are three options defining what can be included from the respective table in a solution, as illustrated in the following screenshot:



Figure A.12: Segmentation options

It is important to understand what these three options do exactly and the best practices for using them:

- **Include all components:** This will add all components (metadata, columns, relationships, forms, view, charts, and others) of a table into the solution package. This option is used by default when creating new tables within a solution package. For existing tables, this option should only be used if that table is not yet present in the target environment.
- **Include entity metadata:** This will add only the table with its metadata, which is a collection of basic and behavioral settings for the table, but no table components. This option should only be selected when there is a need to change some of the table settings in the target environment.

- **Select components:** This is the true solution segmentation capability because clicking on this link opens a dialog where any table components can be individually selected and included in the solution. This option should be used for every situation when a modification of a particular component of a table should be deployed, for example, solution patching or solution updates.

## Using source control

The Power Platform solution package is a ZIP file, which is technically a binary file. Although it is certainly possible to commit this type of artifact into a repository, it would not be possible to use it within the repository for any further operations, such as content comparing. That's why it is highly recommended to always use the unpacking capability of the **Package Deployer tool** or the respective **Azure DevOps task** to first unpack the solution package into its components and commit the components into the repository. The solution package contains, for example, the whole customization configuration as a group of XML files and this can be very useful for analyzing customizations, comparing customization versions, and similar activities.

## Use one single publisher for all solutions within a project

In the previous sections, we discussed several potential issues with collisions, dependencies, and so on, and presented various best practice approaches. The single most important best practice for the solution publishers to make all of the other best practices work is to always use **one single publisher for all solutions within a project**, regardless of how connected or isolated parts of the overall developed solution are.

## Power BI best practices

When developing Power BI components as part of an overall Power Platform solution, the following best practices can help to achieve more governance and consistency:

- Separation of datasets and visualizations, where the datasets can be created by data experts knowing the structure of the data model of the application, while the visualizations (reports and dashboards) can be created by more business-related experts
- Carefully structuring the required datasets to limit the amount of processed data and the size of the datasets
- Using pre-created templates for reports and dashboards (**PBIT** files) to ensure visual consistency of the created content
- Preferably using the **PBIT** files for ALM since they do not contain any data and are much smaller in size

- Using Power BI parameters in datasets to allow easy configuration of separate data sources for development, testing, and production

## Security best practices

In *Chapter 7, Microsoft Power Platform Security*, we cover all the security possibilities when designing Power Platform solution architecture, including authentication, authorization, compliance, privacy, and data protection. In this section, you will make yourself familiar with some best practices for building a robust and mature Power Platform security model. Following the best practices can ensure that your Power Platform solution fulfills the necessary security standards and that you are building a solution that can be maintained and upgraded in the future.

### Dataverse security roles

In this section, we will present some proven best practices for designing, implementing, and using Dataverse security roles.

#### Create new custom roles instead of modifying default roles

Security configuration in Dataverse applications can be very simple, where the default Dataverse or Dynamics 365 security roles can be used without any modification. However, if the security requirements are advanced, there might be a need to tailor the permissions provided in the standard roles. In such a case, it is highly recommended **not to modify** the default security roles, but rather create new custom roles. The best method is to select the best suitable standard security role, make a copy of it, modify the permissions as needed, and then use the custom role instead of a standard role.

#### Layer security roles instead of configuring individual roles

Every user in a Dataverse-based application must have at least one security role assigned, but there is no restriction on how many security roles can be assigned to an individual user.

Configuring individual security roles for every required permission combination is not always the best way, since it might require creating a multitude of similar security roles, which could make it challenging to maintain. An alternative approach to a situation where there are many required permission combinations and most of the standard security roles cannot be used is to use security roles layering, as illustrated in the following diagram:

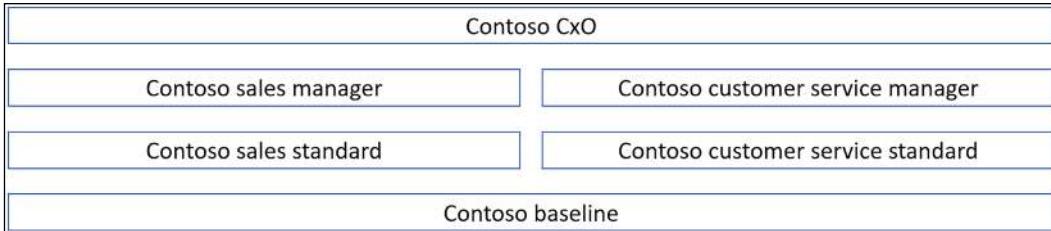


Figure A.13: Security roles layering

As we can see, every user in the organization will be assigned a certain number of security roles with additive permissions, based on their level in the organization. The configuration in this example is as follows:

- **Contoso baseline:** This security role will specify basic permissions for every individual user in the organization.
- **Contoso sales standard:** This security role will specify additional permissions for every sales representative in the organization.
- **Contoso customer service standard:** This security role will specify additional permissions for every customer service representative in the organization.
- **Contoso sales manager:** This security role will specify additional permissions for every sales manager in the organization.
- **Contoso customer service manager:** This security role will specify additional permissions for every customer service manager in the organization.
- **Contoso CxO:** This security role will specify additional permissions for the members of the top-level management of the organization.

This approach is beneficial in different ways. These benefits are as follows:

- Every individual security role above the baseline specifies only the additional permissions against the baseline or lower-level security roles. This ensures that the security roles' configuration is easier compared to the standard security roles approach, where every role contains the full set of permissions needed for the role.
- If a general permission change is required, this change can be performed in a single security role instead of several of them.
- Generally, this approach would require fewer security roles to be created and maintained.

In case the permission requirements are only slightly different from the standard security roles, it can be beneficial to keep using the standard security roles, but to also create one or a few additional custom security roles that extend the permissions of the standard roles.

## **Dataverse content-based security**

All the Dataverse security models described in *Chapter 7* are user-centric, which means the privileges that are granted are always inherited from the user accessing Dataverse. There are, however, situations where content-based security could be required. A typical example could be a Dataverse solution for a bank, where all private customer records are stored in the standard Contact table. Let's assume 98% of the private customers are usual customers and the last 2% are VIP customers. While the data of the usual customers can be seen by every user within the standard role-based authorization model, for the VIP customers, certain data should be available to only a small group of users.

There are several possibilities to solve this content-based security requirement, as described in the following sections.

## **Using a business unit hierarchy**

Using business units is the simplest approach as we only leverage standard Dataverse capabilities. The essence of this solution would be to create an appropriate business unit hierarchy and offload the ownership of the sensitive business records to a user or team, assigned to a business unit that can't be accessed by the vast majority of the Dataverse application users.

The benefit of this approach is that it uses standard Dataverse capabilities and that access to the protected data is reliably restricted in all parts of the Dataverse solution.

The drawback of this approach, however, is that all the data are hidden from the standard users, not just the sensitive parts.

## Using table form switching

Table form switching is a frontend-only solution that leverages the capability of having more than one main form per table. Instead of managing access to table forms based on security roles, this approach would switch table forms based on content during the onloading event using JavaScript event handling.

The benefit of this approach is the selective show/hide capability of sensitive information based on content, by just configuring the table forms with exactly the required content.

The big disadvantage is that this approach is not 100% safe, since skilled users can figure out how to get access to the hidden pieces of information using the advanced find capability, or other similar approaches.



Client-side scripting or business rules can to some extent achieve the same as table form switching with all the benefits and disadvantages.

## Using server-side event handlers

Using server-side event handlers is a backend solution that leverages the capability of creating *Dataverse plug-ins*. The essence of this solution would be to implement server-side event handlers that hide/mask some sensitive data based on the content of the business record. These event handlers would need to be registered for both the **Retrieve** and **RetrieveMultiple** Dataverse events to protect the sensitive data in both table views as well as table forms. This approach requires custom development and protects the sensitive data reliably across all parts of the Dataverse solution, except for the standard Dataverse reports based on Microsoft **SQL Server Reporting Services (SSRS)**.

In the following section, we will discuss some additional ways to achieve a consistent authorization solution across various cloud components used within a Power Platform solution.

## Integrate security across solution components

A Power Platform solution usually consists of several Power Platform and other components, from which some are directly integrated at the level of individual Dataverse business records. The best example of such a solution would be a Dataverse solution that's been extended with SharePoint for integrated document management and Power BI for analytics and reporting, as shown in the following diagram:

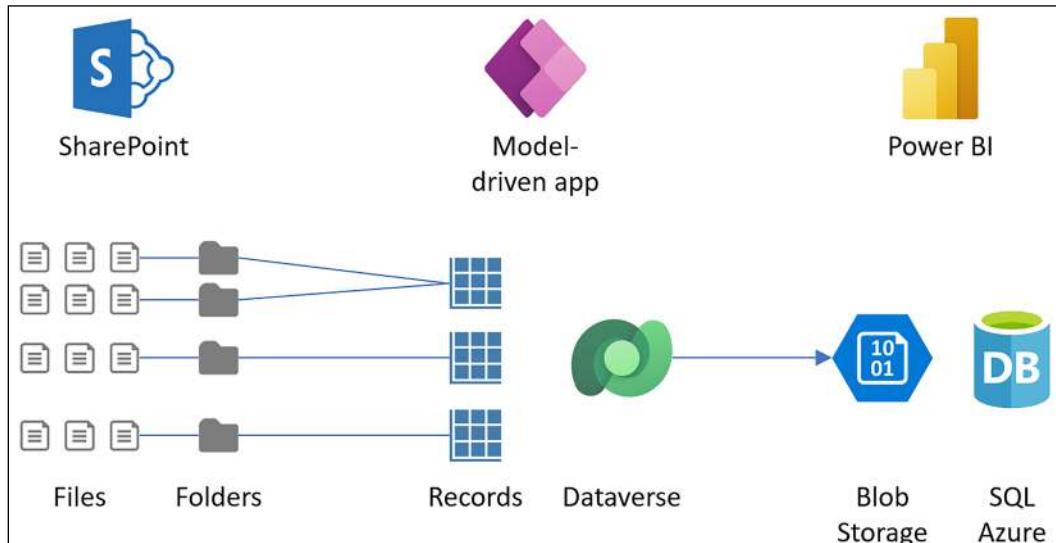


Figure A.14: Power Platform solution example

As we can see, the standard integration approach works as follows:

- For every SharePoint-enabled Dataverse table, there can be one or more SharePoint folders where users can upload files and work with them from within the context of the Dataverse record in a model-driven app. For this standard integration to work, every Dataverse user would need to have proper permission on the whole **SharePoint** site collection, integrated with Dataverse.
- A standard Power BI solution for Dataverse performs an import of the selected Dataverse records into the Power BI internal storage (**Azure Blob Storage** and **Azure SQL Database**) and uses this local copy of the Dataverse data as a dataset for analytics and reporting. The dataset is equipped with the credentials of the creator, who has full permission to all the records in the Dataverse table.

Both of these standard solution approaches do not respect the very detailed authorization concept of the Dataverse applications and, instead, use their own not-integrated authorization concept for accessing data – folders and files in SharePoint and business data in Power BI reports and dashboards. The security consequence of not implementing any additional security solution would be as follows:

- Every Dataverse application user would be able to switch over to the integrated SharePoint site collection and find any files belonging to any Dataverse records they have no permission to see within Dataverse.
- Every Dataverse application user would see all the Dataverse data in the Power BI reports and dashboards, to which the credentials of the creator of the Power BI datasets grant permissions.

If there is a requirement to respect Dataverse-based access permissions to the folders and files in SharePoint and business data in Power BI, an additional layer of security needs to be implemented, as described in the following sections.

## Dataverse-SharePoint integrated security

Achieving a consistent and automated authorization solution for Dataverse and SharePoint is possible using some of the following options:

- **Full access:** The standard setup for Dataverse-SharePoint integration requires giving respective contributor permissions to the whole SharePoint site collection, which is used for integration with Dataverse. This is the simplest solution, but it opens up a potential security hole, as described earlier in this section.
- **Manual access control:** This approach would require not granting site collection-wide permissions to all Dataverse users, but rather granting access to individual folders and files in the SharePoint site collection hierarchy manually. This could be done by an administrator. While this is theoretically possible, for a larger number of Dataverse business records, this would be impossible to manage.
- **Permission replication development:** This approach would require building a custom solution for Dataverse to replicate the Dataverse record permissions in the integrated SharePoint folders and files. This would require significant custom development effort to cover all the very specific Dataverse authorization possibilities and their changes over time.
- **Third-party solutions:** There are commercial solutions on the third-party market that cover permission replication from Dataverse to SharePoint.

## Dataverse-Power BI integrated security

When integrating Power BI with Dataverse, there are also some ways to achieve consistency when accessing data:

- **No authorization:** This type of authorization means that Power BI will present all the Dataverse data in the visuals to all users. This is a simple and viable option when providing unfiltered data to everybody is acceptable. This authorization can be implemented using both *import* as well as *DirectQuery datasets*. The only necessary configuration setting is to ensure that the credentials of the datasets allow full access to all the data in all the tables being used in the Power BI visuals.
- **Static authorization:** This option is a modification of the previous one. We would need to specify dataset credentials that do restrict access to some of the Dataverse data, but the result would present the same statically pre-filtered subset of data to every user in the Power BI visuals.
- **Row-level security authorization:** This type of authorization would require a parallel authorization level to be built along the Dataverse authorization. This can be an intended approach when the Power BI visuals should present data that's been filtered using different criteria than what is implemented in the Dataverse authorization. This authorization can be implemented using both *import* as well as *DirectQuery datasets*. The configuration would require the use of credentials for the datasets that allow full access to all the data in all the tables being used in the Power BI visuals, as well as creating the respective RLS roles and assigning them to Power BI users.
- **Dataverse authorization:** This type of authorization can be implemented using the *DirectQuery* dataset type only. The Power BI visuals would then display the same data to the Power BI users that they can see within Dataverse applications. The only necessary configuration setting is to ensure that the credentials of every individual user will be propagated to the data query when data is being retrieved from Dataverse.

In the following sections, we will discuss the topic of automating the whole identity and access management process once more, but this time, we will include the ability to integrate into an on-premises active directory.

## How to use identity and access management automation

Large organizations with complex IT ecosystems usually use **identity and access management (IAM)** solutions to provision new users and manage user permissions within existing IT systems. It is important to have the capability to automate these processes for complex Power Platform-based solutions as well, to be able to support the existing customer's IAM solution.

This topic was already touched upon in *Chapter 3, Understanding the Microsoft Power Platform Architecture*, so now we will extend this to scenarios that cover Active Directory (AD) integration. For all AD integration approaches, there is an additional component called **AAD Sync** (part of **AAD Connect**) that's deployed. This synchronizes the AD user accounts in AAD. An IAM automation solution would need to consider this feature and work differently compared to a solution for pure cloud identities, as illustrated in the following diagram:

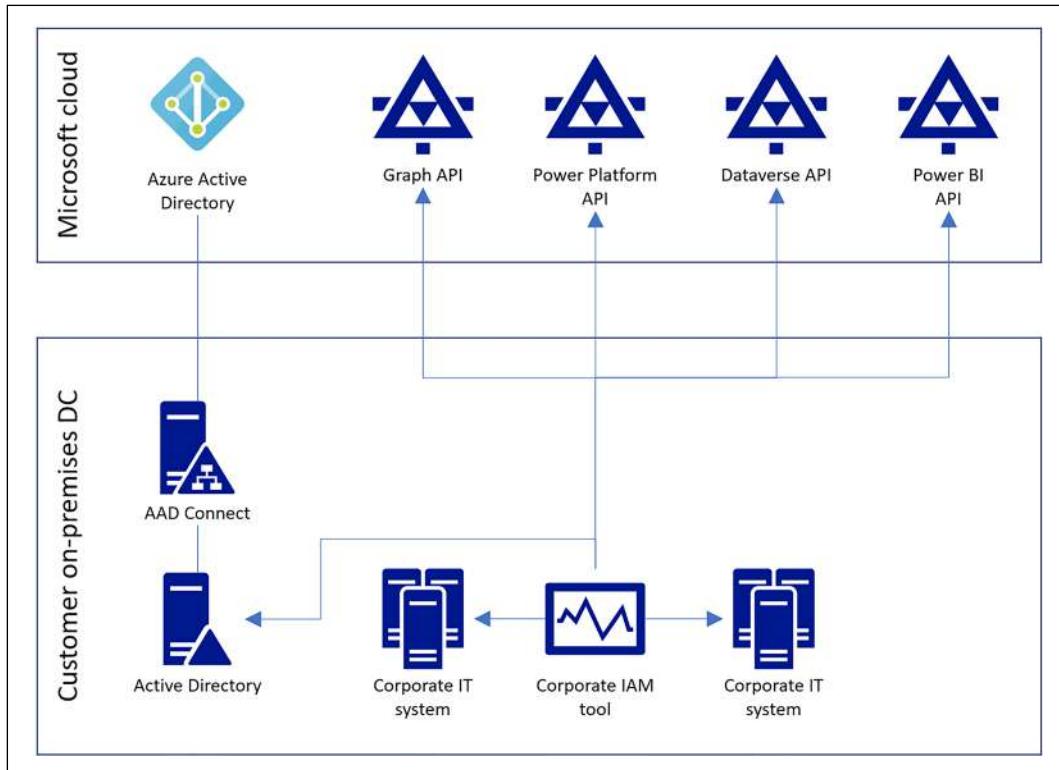


Figure A.15: Identity and access management automation

As we can see, the first step in provisioning a new cloud user account in AAD is now fully under the control of AAD Connect, specifically the AAD Sync service. The AAD Sync service is a service that's triggered with a timer, usually every 15 minutes, which then performs all pending user account synchronizations. To fully automate the IAM process for Power Platform, we need to follow these steps, which need to be implemented within the customer's IAM solution:

1. Create the new user identity in the on-premises active directory and flag the new user account as cloud-enabled. After this, AAD Sync will recognize this new AD user for synchronization in AAD.

2. Start a polling sub-process to poll AAD using the *Graph API*, to figure out when the user account is already provisioned in the AAD.
3. OPTIONAL: After the new user account has been provisioned in AAD, assign the user to all necessary security/Office 365 groups associated with the Dataverse environments where the user should be granted access.
4. Assign the necessary Power Platform licenses to the new user using the *Graph API*.
5. Since provisioning the new user in the Dataverse environments is also an asynchronous process, start a polling sub-process to poll all the Dataverse environments using the *Dataverse API*, to wait for the automated user creation. Alternatively, create the users in Dataverse actively, using the *Dataverse API*.
6. In every Dataverse environment where the new user is already provisioned, assign the necessary authorization privileges and permissions using the *Dataverse API*.
7. Assign the necessary authorization privileges in Power BI using the *Power BI API*.

## Establishing a Power Platform mature security model

With the help of the security concepts of Power Platform and its components, it is possible to establish a mature security model at all possible levels, as illustrated in the following diagram:

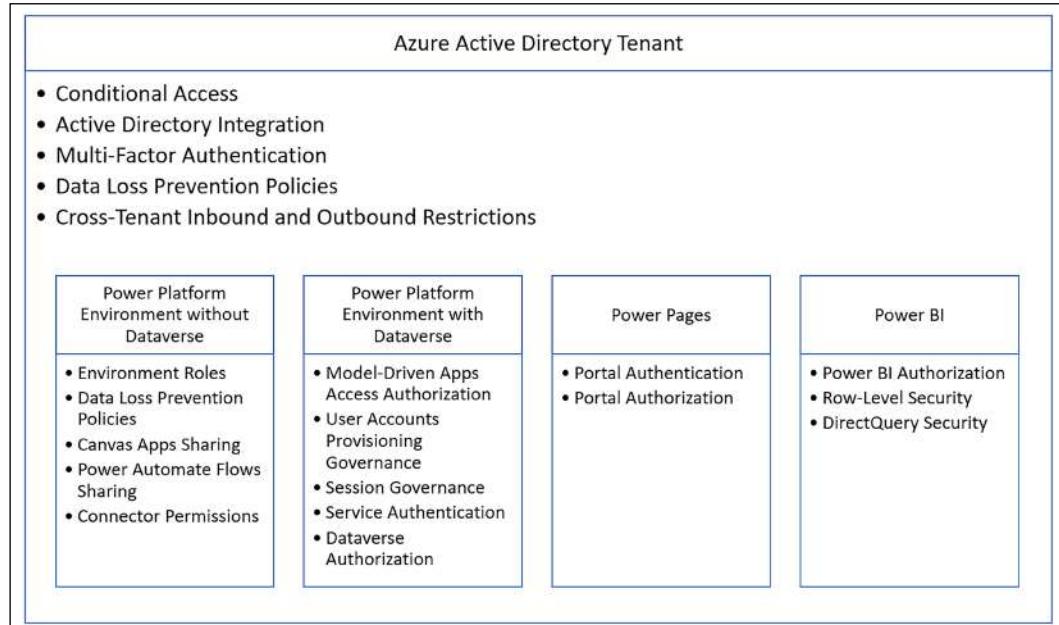


Figure A.16: Mature Power Platform security model

As we can see, there are certain security options available on the whole **Azure Active Directory Tenant**, and others are available within the different Power Platform components. The various options can be used for the following purposes:

- **Conditional access:** Use AAD conditional access if you need to generally restrict access to the whole AAD tenant and all the components within the tenant, including Power Platform, for users based on defined criteria (geography, device, application, and membership).
- **Active Directory integration:** Use Active Directory integration to ensure centralized password control, centralized identity policies, and single sign-on.
- **MFA:** Use MFA to increase the security of the authentication process and prevent malicious authentication attacks.
- **Data loss prevention policies:** Use data loss prevention policies to prevent unattended business data leakage through Power Automate flows and canvas apps.
- **Cross-tenant inbound and outbound restrictions:** Use cross-tenant inbound and outbound restrictions to prevent the use of foreign public cloud services by your own users or your own public cloud services by foreign users.
- **Environment roles:** Use environment roles to manage proper access to environments and environment permissions.
- **Canvas Apps sharing:** Share canvas apps properly within the organization.
- **Power Automate flows sharing:** Share Power Automate Flows properly within the organization.
- **Connector permissions:** Ensure proper use of connector permissions to provide every user access to only the necessary systems and data.
- **Model-driven apps access authorization:** Use model-driven apps access authorization to provide every user group within the organization access to only the necessary model-driven apps.
- **User accounts provisioning governance:** Use user accounts provisioning governance to avoid provisioning user accounts into unwanted Dataverse environments.
- **Session governance:** Use session governance to prevent the misuse of Dataverse applications by unauthorized users and to enforce policy changes, which are applied during the user authentication process.
- **Service authentication:** Use the proper service authentication model when developing external applications that connect to Dataverse or Power BI environments.

- **Dataverse authorization:** Plan, design, and implement a proper Dataverse authorization model to ensure every user group has adequate permissions within Dataverse applications.
- **Portal authentication:** Choose proper portal authentication options to enable external users to authenticate with their favorite identity providers.
- **Portal authorization:** Plan, design, and implement a proper portal authorization model to ensure every external portal user group has adequate permissions within the portals.
- **Power BI authorization:** Implement proper Power BI authorization to enable access to all necessary Power BI components for every user group.
- **Row-level security:** Use row-level security within Power BI to manage access to Power BI data for different user groups based on role permissions.
- **DirectQuery security:** Use DirectQuery security to apply the authorization model of the underlying data source within Power BI.

## Extensibility best practices

In *Chapter 8, Microsoft Power Platform Extensibility*, we learned how we can extend the abilities of Power Platform components using configuration, customization, and custom development. As you might recall, we demonstrated that Dataverse is the most extensible complex Power Platform component. In the following sections, we will go over some of the best practices for client-side as well as server-side extensibility for Dataverse-based solutions.

### Optimizing the performance of client-side extensibility

It is always important to keep the performance of the Dataverse-based solution in mind. Client-side extensibility options can have a significant impact on the final solution's performance. Try to implement the following best practices when designing the extensibility to avoid the most common sources of poor performance:

- Design the Dataverse table forms by providing only the necessary controls; avoid overwhelming the forms with many complex controls. Very large forms usually load slower, which obviously leads to poor acceptance.
- Use more tabs with a smaller number of sections and controls, rather than many sections and controls on a single tab. Using this approach can help the end user start working with the data on the form faster, without the need to wait for all the content on the tab to load.

- When using embedded IFrame form components, consider loading the content asynchronously to speed up the form's load time. This can be done using some advanced JavaScript code. An easier way to handle IFrames is just not to place them on the first form's tab so that the content can load in the background, while the end user is working on the most important data.
- Carefully consider using embedded canvas apps on Dataverse forms for performance reasons – do not place canvas apps on the first form tab.
- Use PCF controls rather than web resources, since they load fast in the main form load event.
- Use Dataverse business rules instead of JavaScript event handlers whenever possible. Business rules are an integral part of the Dataverse platform and usually perform faster than JavaScript event handlers.
- Consider the most performance-sensitive client-side events, such as OnLoad, and avoid registering complex JavaScript event handlers for those events. Complex and long-running JavaScript event handlers can significantly slow down the form's load performance.
- For any communication with the Dataverse API, use the client-side Web API instead of SOAP-based calls. It is faster and much easier to implement.
- Avoid using multiple calls to several external web service endpoints in the JavaScript event handlers, especially for performance-sensitive client-side events. Having multiple calls in an event can significantly slow down performance since every call adds up additional time for sending the request, server-side processing, and receiving the response. For high-latency scenarios, this can have an extremely negative performance impact. Consider developing a specific wrapper web service to consolidate the whole server-side business logic into a single service that can be called from JavaScript instead.

## **Dataverse server-side extensibility**

There are many possibilities for the server-side extensibility of Dataverse. In the following sections, you will learn how to select the most suitable Dataverse APIs, how to select the best extensibility and automation options, and what should be considered when building a performance optimized Dataverse solution.

## When to use which Dataverse API

When planning to build Dataverse server-side extensibility, there are three possibilities regarding the use of APIs and tooling:

- **Plug-ins and custom workflow actions:** For these types of extensibilities, the only option is to use the organization service, which is part of the execution context. The benefit of doing this is having the respective DLL assemblies available, which makes the development process and the existing pre-authentication of any communication easier.
- **External .NET-based applications:** For these types of extensibilities, the most suitable approach is to use the **XRM Tooling** assemblies, since they provide a set of DLL assemblies with a lot of ready-made capabilities for custom development. For example, the whole authentication complexity is covered by one of the classes contained in XRM Tooling.

It is recommended to use OAuth authentication for increased security instead of Office 365 authentication, even though Office 365 authentication is available within XRM Tooling.

Another option, when not using the XRM Tooling assemblies, is to resort to the modern Web API, OAuth authentication, and either the standard .NET HTTP client or any of the third-party supporting libraries.

- **External application, not .NET based:** For these types of extensibilities, the only option is to use the Web API, since Microsoft does not provide any support for SOAP from non-.NET solutions.

Always select the proper authentication method against the Dataverse API. The use of legacy Office 365 authentication is possible for SOAP-based endpoints; however, it is considered less secure. Use OAuth whenever possible, even for SOAP.

### Important note



There are good community sources available that provide supporting information for various frameworks and programming languages for using the Web API against the Dataverse API. Please refer to the *Further reading* section at the end of this chapter for more information.

In the next section, we will present a comparison between the various approaches for using extensibility and implementing automations.

## Recapping extensibility and automation options

In *Chapter 8*, you have learned a lot about various server-side Dataverse extensibility and automation options. In this section, we've provided a summary and overview of the various options available, highlighting the typical usage scenarios and technology constraints:

- **Dataverse business rules:** Simple automations based on a single table record. They can replace JavaScript client scripting to some extent, and they also work on the server side.
- **Dataverse workflows:** Medium-complexity automations that can only run on the server side and can execute synchronously or asynchronously. For asynchronous scenarios, these can be replaced with Power Automate.
- **Dataverse custom actions/custom APIs:** Medium-complexity automations that can only run on the server side. They do not have their own triggers, so they need to be triggered from a Dataverse workflow, business process flow, or code.
- **Dataverse business process flows:** Simple user interface-based automations. They provide guidance to end users in terms of following standard business processes. They can include various embedded automations using Dataverse workflows, Dataverse custom actions, or Power Automate flows.
- **Power Automate:** Medium-complexity automations that can only run on the server side and only asynchronously. It has a broader range of capabilities compared to Dataverse workflows. It can run as an event handler but can also be triggered manually from the Dataverse application's command bar or from Dataverse business process flows.
- **Plug-ins:** High-complexity automations that can only run on the server side and only as event handlers; they cannot be triggered manually. They can be implemented for synchronous or asynchronous execution and need to be developed using custom code in .NET. Plug-ins should be considered when any of the low-code/no-code approaches are unable to cover complex business requirements.
- **Custom workflow actions:** High-complexity automations that can only run on the server-side and must be triggered from Dataverse workflows. They need to be developed using custom code in .NET. The usage scenario is the same as it is for plug-ins. The difference is that using custom workflow actions provide the flexibility needed to change the base business logic without code within the Dataverse workflow, while for plug-ins, every change in the business logic must be implemented by modifying the code.
- **Azure Service Bus integration:** Can be used to implement remote automation or outbound hybrid integration scenarios in asynchronous mode only. Requires a separate Microsoft Azure license. The listener component needs to be developed using custom code in .NET.

- **Webhook integration:** Can be used to implement remote automation or outbound integration scenarios in synchronous or asynchronous mode. The functionality can be implemented in the cloud, but also on-premises.
- **Building external applications:** Can be used to implement a variety of business requirements from batch jobs, through enterprise integration scenarios, to every kind of alternate client application solutions.

In the next section, you will learn about some of the best practices in terms of performance optimization for Dataverse applications.

## Avoid synchronous workflows and plug-ins

When architecting and designing server-side extensibility, it is very important to keep the performance of the future solution in mind the same way as you would for client-side extensibility. There are certain solution approaches that should be limited or entirely avoided since they have a known negative performance impact:

- **Synchronous Dataverse workflows** should only be used when the requirement dictates to have the result of the workflow being processed available immediately on the user interface. It is recommended to reduce the complexity and duration of synchronous workflows to a minimum.
- The same recommendation applies to **synchronous plug-ins**.
- Implementing business logic using plug-ins registered for the `Retrieve` or `RetrieveMultiple` events should be considered very carefully. Such plug-ins are known to have a serious negative performance impact. If required, performance optimization of the plug-in's code is necessary.

## Integration best practices

In *Chapter 9, Microsoft Power Platform Integration*, we covered how to integrate Power Platform solutions into an existing IT infrastructure. Learning about these approaches involved gaining an awareness of which integrations are easy to do, and which require additional components and custom development. Using some of the integration patterns and solution approaches we described is certainly useful, but not using them in the best way can cause various issues, especially regarding the solution's overall performance. In this section, we will present some best practices for frontend and backend integration.

## Frontend integration

When deciding on which frontend integrations to use, there are some important best practices that should be followed in order to achieve a robust, maintainable, and performant solution:

- When embedding third-party content into Dataverse table forms, consider placing it on separate form tabs, not on the primary tab, to avoid waiting for the content to load.
- While jQuery is a very handy JavaScript library, it is not recommended to use it to call the Dataverse API. Use the *client-side Web API* to make calls to Dataverse, and only use jQuery to call external web services.
- For any server-side calls, consider always using *asynchronous communication*. Developing asynchronous communication is more complex, but the result is a smooth end user experience without unexpected delays and the user interface freezing. Using the XMLHttpRequest() method in a synchronous way is already deprecated and will be removed in future versions of the main browsers.
- Whenever possible, prefer on-demand frontend integration over event-driven integration. When frontend integration is implementing communication with third-party IT systems, it is fully dependent on the availability and performance of the external endpoints. Having this communication triggered by the end user gives the user more control and avoids unexpected delay or other abnormal situations.

In the next section, you will learn about some backend integration best practices.

## Backend integration

When performing backend integration there are several opportunities to leverage some best practices that can lead to simpler code and faster, more reliable solutions.

The following are the best practices for backend integration:

- All the common *update* and *delete* methods of the Dataverse API require the use of the *primary key (GUID)* of the table record to identify the record for update or delete. However, in integration scenarios, the primary keys of the Dataverse records are, in most cases, unknown by the integrated third-party solution or system. A good way to overcome this limitation is to hold the known record identifiers from the integrated systems as columns in the Dataverse tables (for example, customer number, product number, or case number) and configure those columns as **alternate keys**. Using alternate keys in integration scenarios makes it possible to use these known identifiers of the records in the update or delete methods instead of the primary keys, and it also saves an additional retrieve request to find the primary key first.

- In data integration scenarios, there is often a situation where some of the records coming from the third-party IT solution are not available in Dataverse yet, but some are already stored in Dataverse and just need to be updated. To simplify the solution, there is now the possibility to use the **UPSERT** request, which is part of the Dataverse API. An UPSERT request can automatically detect whether a record already exists in Dataverse. For existing records, there will be an automatic UPDATE transaction, while for non-existent records, a CREATE transaction is executed.
- To ensure there's a certain level of transactional safety and to pack multiple individual data modification requests into a single technical API call, it is possible to use **batch requests** when calling the Dataverse API. A batch request can contain up to 1,000 individual requests.
- For pure outbound integration scenarios, consider using the new Dataverse API endpoint known as **Tabular Data Stream (TDS)**. At the time of writing, this endpoint can only be used for reading data from the Dataverse database, but skilled SQL developers will quickly find it very useful for designing complex queries against Dataverse using the SQL language. The TDS endpoint is also very useful for checking Dataverse data using *SQL Server Management Studio*, or to use it for building Power BI reports and dashboards.

## Data migration best practices

In *Chapter 10, Microsoft Power Platform Data Migration*, we explained that performing the data migration that must accompany the solution integration is one of the biggest challenges in an enterprise solution project. As part of the overall project, it is often underestimated and not planned properly. There are a lot of challenges that need to be considered and included in the project planning and execution phase, as described in the following sections.

### Don't underestimate the project duration

One of the most observed challenges for data migration projects within Power Platform implementations is *underestimating* the overall effort, project duration, complexity, and timing. Data migration for large enterprise solutions is always a separate project that must be carefully planned and started at the beginning of the overall project undertaking. This helps ensure the migration solution is ready at go-live time.

### Determine the scope of the migration

With a large Power Platform implementation, in the beginning, it is often required to migrate every possible data domain into the solution so that you have Power Platform as a centralized data hub. This is, in most cases, not the correct way to understand a Power Platform solution; the solution is neither a data warehouse nor a data master for every piece of information in the organization.

Therefore, it is important to carefully analyze the data architecture of an organization to understand the expectations, as well as to decide which role the Power Platform solution will play in the overall enterprise architecture of the organization. In other words, **scoping a migration** is a very important part of the planning and design phase.

To cover, for example, the organization's requirements for a 365-degree customer view, a lot of secondary data elements do not need to be physically imported. It is sufficient to visualize them using solution approaches such as *virtual tables* or embedded *HTML web resources* or *PCF components*. The best practice to decide the scope of the data migration is to classify the data tables on whether they need to be editable, searchable, and involved in business processes in the Power Platform solution. If the answer is no, then that data table can probably be excluded from the physical migration.

Another part of scoping is to decide whether all the records from a selected data table need to be migrated or only a subset, such as those created in the last year or in the last 3 years. Older records can be considered archived and do not need to be migrated.

A good example of a bad migration design is to try to physically migrate retail transactional data into Dataverse. In industries such as retail, travel, banking, and others, every customer can generate hundreds or even thousands of individual retail transaction records. While the data itself is very interesting for a Power Platform solution, there is no reason to import that data physically into Dataverse due to the huge data volume. A much better solution would be to either visualize the data using one of the approaches mentioned previously, like virtual tables, HTML web resources, or PCF components, or to migrate aggregates from that data based on proper grouping and clustering.

## **Understand the impact on storage**

The agreed scope for data migration has an immediate impact on the storage requirements for the Dataverse solution. As you learned in *Chapter 3, Understanding the Microsoft Power Platform Architecture*, Dataverse comes with a certain default storage capacity that depends on the number of purchased Power Platform or Dynamics 365 product licenses. It is important to perform a high-level storage requirement analysis. This helps to assess whether the available storage space will be sufficient to accommodate all the data that's planned to be imported. At this point, you may also decide to purchase storage capacity add-ons to mitigate possible storage bottlenecks.

## Compliance considerations

Another follow-up from scoping the migration process is to discuss **compliance**. This helps to assess whether all the data that's been planned to be migrated to Dataverse can be physically stored in a public cloud. There might be a situation where, due to some governmental or other regulations, certain data elements are not allowed to be physically stored in a cloud.

## Start getting physical access to all required systems and solutions early

Data migration often needs various data sources from legacy solutions running in the customer's own data centers or hosted by various hosting partners. For the data migration project, it is essential to get *physical access* to all systems and solutions containing relevant data sources. However, this can often be a lengthy process that requires complex approvals, security settings, access rights, and permissions to be provided, and so on. It is recommended to identify all the source systems early in the project stage to ensure access to those is granted early enough for planning, designing, developing, and testing the migration procedures.

## Expect a lack of knowledge about legacy IT systems

Data migration must often be performed from very old legacy IT systems that are still running in production, but where *limited insider know-how* exists in the organization regarding the technical details of the solution. Since the data structures of legacy systems can be extremely complex and hard to understand, it is imperative to ensure subject matter experts are available for the project.

On a similar note, technical documentation for legacy IT systems is sometimes *not available* or *outdated*, thus not reflecting the current state of the solution. It is important to try to obtain as much technical documentation for the systems as possible. This helps us understand the technical details and the data model of the solution. Documenting your own data migration solution is, of course, part of the migration project.

## Include contractual responsibility for the quality of source data

It is almost a tradition for the quality of the data in the source systems to be poor and not to correspond with the high-quality data standards in Power Platform solutions. For a Power Platform implementation team, it is difficult to improve the data's quality for the migration process. The following are the typical data quality issues we have observed in data migration projects:

- Duplicate records (this issue occurs even more when consolidating data for the same data domain from different data sources)
- Mandatory fields are empty
- No unique record identifiers
- No clear relationships between data tables
- Missing record ownership information

The best practice for this situation is to make the customer contractually responsible for the data cleansing process and for the overall quality of the source data upfront. The data cleansing process can technically be performed at three different stages in the migration: in the source IT systems, to make the whole migration process easier; during the migration process, if there is an opportunity to do that; or after the migration process in the Dataverse solution itself. However, experience shows us that when proper data cleansing is not performed in the source systems before the data migration happens, the customer is seldom able to perform real post-import cleansing.

## **Legacy IT systems might have encoding issues**

Issues with encoding are rare today since most of the current IT solutions and systems out there support **Unicode data encoding**, but when planning data migration from very old legacy IT systems, this might be an additional issue. Power Platform requires all data to be encoded as *Unicode* to support the broad variety of languages being used in the world. Therefore, it might be important to perform an additional encoding step to ensure all the data is really in Unicode.

## **Attempt to resolve record ownership issues in the data transformation phase**

One of the specific issues when migrating data to Dataverse is the *ownership* of records. Records in every table with user/team ownership type must have an *owner*. The owner of the record is one of the most important data elements for the whole Dataverse authorization process to work properly. If the source data does not contain any useful information about the possible owner, the data must be imported in the security context of a predefined owner and the correct assignment must be performed after the migration. However, it is much better to resolve this ownership issue during the migration process. It is recommended to, when possible, include the ownership information in the source data during the transformation phase of the ETL process.

There is another specific issue that needs to be resolved if there is certain ownership information in the source data but many of the owners of the records are no longer in the organization or do not plan to use the Dataverse application. For this specific use case, **stub users** can be leveraged, as explained in *Chapter 7, Microsoft Power Platform Security*.

Another important topic to evaluate is the use of a certain user or service account, under which the migration solution should work. This account will need to be provisioned in the Dataverse solution and equipped with all the necessary access rights. This account will be the owner of every data record for which no specific owner is explicitly defined. In this context, the *impersonation* capability of the Dataverse API can be considered.

## Understanding mapping issues

One of the main parts of the data migration process is the mapping of the data between the source and Power Platform and specifically the Dataverse data structure. The mapping for simple data types such as **string**, **number**, **date**, and **time** is usually simple, but mapping for Dataverse-specific data types such as **choice**, **image**, **customer**, or **file** can be more challenging.

Data mapping generally consists of the following steps:

1. Map the data at the **table** level (decide which source data table maps to which Dataverse table).
2. Map at the **column** level (decide which columns from the source table map to which columns in Dataverse).
3. Map at the **column values** (this is specifically important for columns, where the possible values are limited, such as choice, lookup, etc.).

One of the typical issues is to map the data type *choice* correctly. As we know, the choice consists of an internal numerical value and a text representation of that value. For correct mapping, the choice values must be imported from Dataverse to the staging database to be available for mapping before the main migration can be even performed.

Another important consideration is to understand that *datetime* fields are physically stored in the Dataverse database in UTC format. For performance optimization reasons, it is recommended to set the time zone of the user or service account performing the migration to UTC as well. This will avoid time zone conversions during data loading. It is also very important to understand the datetime formatting in the source IT systems, which might not necessarily be in the expected time zone. Here, a conversion would need to be performed.

## Exclude records with record relationship issues

The Dataverse database requires that we create a consistent data model including all relationships between tables. However, data in source IT systems is not always stored in database systems and respects the same level of consistency between related data tables. Simply put, parts of the records in expectedly related tables do not have the connection between them established. This issue is extremely hard to resolve, and, in most cases, these records just need to be excluded from the migration process.

## Understanding business process flow issues

In some specific situations, you will need to assign **business process flows (BPFs)** to imported records. This can happen for records in the typical main business tables, such as *opportunities* in Dynamics 365 Sales or *cases* in Dynamics 365 Customer Service. The real challenge is not assigning the BPF itself but assigning the required stage, since the required target stage could be any of the existing stages. The typical procedure to achieve the desired BPF stage for an imported record is illustrated in the following implementation example.

The requirement is to import opportunities from a legacy CRM system into Dataverse. Each of the legacy opportunities has an equivalent of a BPF stage, where there could be stage values between 1 and 6. The implementation needs to contain the following steps:

1. Import a new record into the **Opportunity** table.
2. Apply the required BPF to the new record.
3. Perform a forward stage change until the desired stage value is reached. (This step can be performed between 0 to 5 times over).
4. Optionally close the opportunity as **Won** or **Lost**, according to the business requirements.

As you can see from this example, the whole process can consist of up to eight different transactions issued against the Dataverse API. This illustrates the complexity of certain data migration requirements.

## Understanding record status issues

Every Dataverse record has *status* and *status reason* columns by default. The *status* column defines the major behavior characteristics of the business record. Any status that corresponds to the generic status of **inactive** (it can be inactive itself or **Won/Lost** for opportunities) makes the business record read-only. This can be an issue for scenarios where it is required to import records that are designated as inactive and then make changes to the records and/or create records in the underlying tables. As an example, we could consider importing old opportunities that are mostly closed together with opportunity products. The correct sequence of import steps is as follows:

1. Import an opportunity record as open.
2. Make all the required updates to the opportunity record; for example, assign a BPF to the record.
3. Import opportunity products for the opportunity record.
4. Move the opportunity's BPF to the required stage.
5. Close the opportunity as **Won** or **Lost**, as required.

As we can see, the combination of BPF and status values can make the import process for certain tables pretty complex. This is because it consists of many individual steps in order to respect all the specifics of the Dataverse business tables.

## Ascertain whether you need to set certain system fields

Every business record in Dataverse has several system datetime fields indicating the creation date of the record, as well as the date of the last record modification. It is important to analyze the default behavior (columns will be set to the current date of data migration) and decide whether this is the desired outcome. If you need to preserve the original creation dates from the source IT systems, a specific *mapping* must be used. There is a largely unknown column in every Dataverse table that can be used instead of the standard `created_on` column. The name of this column is `overriddencreatedon`. Using this column in the migration mapping process will set the `created_on` column to the required value from the source data.

Similarly, this approach can be used to override the default value for the `created_by` field. You can do this by using the `CallerObjectId` field in the mapping.

## Migrating documents

One of the most frequent requirements for data migration is not just to migrate relational data, but also documents of any type – related or even unrelated to relational data records. Importing documents can be done either by importing them directly into the Dataverse database as annotation records with attachments or into an integrated SharePoint site collection. In both cases, the migration process requires you to identify the related business records and upload the files either to the annotation or to a SharePoint folder, which is assigned to the business record. For the SharePoint-based solution, there is a need to communicate with both the Dataverse API as well as with the SharePoint API in order to achieve the desired result. For migration into SharePoint, a dedicated *SharePoint SSIS connector* from *KingswaySoft* can be used.

## Follow the order of migration steps

A complex data migration process can include dozens or even hundreds of separate import steps into various Dataverse tables. In such a complex case, the order of the import steps is extremely important, due to the dependencies between records in relationships and other data-related considerations. The basic rule is that the import of any referenced data must be performed first. An example sequence of import steps is as follows:

1. **Currency records**
2. **Unit records**
3. **Price list records** (to be assigned to the parent currency)
4. **Product records** (to be assigned to the parent unit)
5. **Price list items** (to be assigned to the parent product, currency, and price list)
6. **Account records** (to be assigned to the parent currency)
7. **Contact records** (to be assigned to the parent account)
8. **Opportunity records** (to be assigned to the parent account and contact)
9. **Opportunity product records** (to be assigned to the parent opportunity and product)

The preceding list is just a simplified example. For large migration projects, the dependency list can be much larger, also containing a lot of custom tables. As shown in the previous example, there is already a hidden relationship issue when we need to set the primary contact for every account. This is not possible only following record creation steps, since the contacts do not exist during account creation. A record update step would need to be inserted into the sequence according to the following example:

1. Account records (to be assigned to the parent currency)
2. Contact records (to be assigned to the parent account)
3. *Updating the primary contact of accounts*
4. Opportunity records (to be assigned to the parent account and contact)

The preceding example illustrates the importance of using the correct sequence of data migration steps to achieve the desired results.

## Understanding migration automation

Since complex data migration usually consists of a multitude of steps – even hundreds of individual steps – the whole migration process must be automated. Every complex migration must be developed and thoroughly tested for the final migration run in the production environment. The number of individual test migration runs can also be considerably high in order to correct every identified mistake in the *migration logic* or *technological* approach. To manage this complexity, you must fully automate the whole migration process to make it easily repeatable and reliable. Automating a complex migration can be achieved using several different approaches, such as using the *SSIS data integration project*, using *PowerShell*, or using *custom development* to call all the required actions with code.

## Understanding migration performance

As you learned in the previous chapters of this book, Dataverse offers certain API endpoints to communicate with. These same endpoints must also be used for data migration purposes since no direct writing access to the underlying Azure SQL database is available. That being said, it is very important to consider the overall data migration duration when planning the final migration before going live. The Dataverse endpoints, Web API, and SOAP provide certain transaction performance that is also limited by the platform API limits. To achieve the best possible migration performance, all these technical limitations must be carefully considered. To maximize the data load performance, the options for using the `ExecuteMultiple` request type and multi-threading need to be used and configured. The best-performing combination of the mentioned options needs to be tested. Finally, to evaluate the final migration performance, full test execution is necessary.

### Important note



For further details about using the `ExecuteMultiple` method in code, please refer to the product documentation: <https://learn.microsoft.com/en-us/power-apps/developer/data-platform/org-service/execute-multiple-requests>.

A specific topic to consider when planning a data migration and preparing the design is to decide which types of automation can be, or must be, disabled during the import's execution. A complex Dataverse solution can have various types of automation implemented (workflows, plug-ins, auditing, duplicate detection rules, Power Automate flows, and others). Some of these automations must be switched off due to business requirements since they should only be executed in normal operation. Some of the automation must be switched off due to the negative performance impacts they might have on record creation and updating transactions.

## Request the lifting of API limits for large projects

As you learned in *Chapter 3, Understanding the Microsoft Power Platform Architecture*, the Dataverse API has some protective limits regarding how many requests a single process can trigger against the API within a defined time period. This is particularly unfortunate for large data migration projects that need to create an expectedly high number of records.

### Important note



For further details about handling the API limits within a data migration solution, please refer to the product documentation: <https://learn.microsoft.com/en-us/power-apps/developer/data-platform/api-limits>.

One possible option to overcome this limitation is to request that these limitations be lifted for the period of data migration. This can be done using the *Microsoft customer support* channel.

## Arrange time for the migration execution

For large migration projects, the time required to execute the full migration can be significantly longer, taking the usual speed of the Dataverse API into consideration, combined with API limits, as described in the previous sections. Therefore, it is very important to test the real duration of the migration, use all possible technical options to increase the migration speed, and set the right expectation for the customer. Large data migrations are usually planned to be performed outside of business hours, on weekends, or during a dedicated migration time. It is very important to ensure that, before starting the final data migration process, all the existing IT systems are stopped so that no new data is generated during the migration process.

If one-time migration is impossible to organize due to operational reasons – certain IT systems cannot be stopped for a longer period of time (several days), for example – it might be useful to think about a multiple-step migration, where the data is migrated in several phases and each phase is therefore shorter in terms of execution time. This approach is very challenging and requires very detailed planning so that you can decide which data in which sequence can be imported in every migration phase.

## **The customer should verify the quality of all migrated data before the final migration**

The last but most important step in the data migration process is to verify the data quality of all migrated data. The final verification process and sign-off must happen before the final migration into production is performed and must include all the required production data. This step is the full responsibility of the customer. The implementation partner must require this sign-off as a prerequisite to trigger the **final migration process**.

With that, you have learned that a complex data migration project can have a lot of challenges and that it needs to be planned and staffed properly. The many practices we have learned should assist with overcoming all these typical challenges.



[packt.com](http://packt.com)

Subscribe to our online digital library for full access to over 7,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

## Why subscribe?

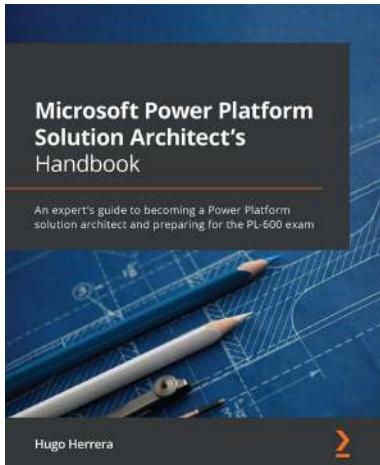
- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Fully searchable for easy access to vital information
- Copy and paste, print, and bookmark content

At [www.packt.com](http://www.packt.com), you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.



# Other Books You May Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:

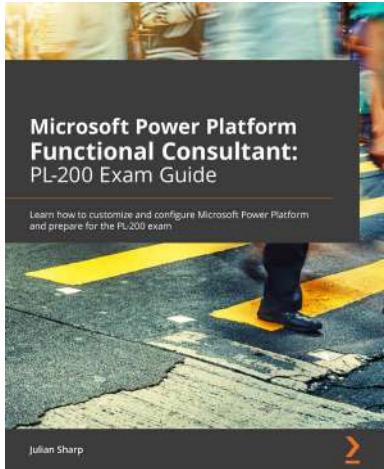


## Microsoft Power Platform Solution Architect's Handbook

Hugo Herrera

ISBN: 9781801819336

- Cement the foundations of your applications using best practices
- Use proven design, build, and go-live strategies to ensure success
- Lead requirements gathering and analysis with confidence
- Secure even the most complex solutions and integrations
- Ensure compliance between the Microsoft ecosystem and your business
- Build resilient test and deployment strategies to optimize solutions

**Microsoft Power Platform Functional Consultant: PL-200 Exam Guide**

Julian Sharp

ISBN: 9781838985684

- Understand how to build apps that meet customer needs
- Extend the schema for Dataverse with entities, fields, and relationships
- Create and configure automations to simplify user activities
- Explore various security features in Power Platform and learn how to implement them
- Use multiple data sources to create task- or role-based web and mobile applications for users
- Automate business processes and enhance the user experience with Power Automate and UI Flows
- Integrate various applications within the Microsoft ecosystem with Power Platform

## Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit [authors.packtpub.com](http://authors.packtpub.com) and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.



# Index

## A

**agent desktop** 319

**agile model** 174

**AI Builder** 13, 113, 381, 382

capabilities 113

characteristics 13

custom AI models 114

prebuilt AI models 114

**Ajax** 363

**Apache Spark**

URL 356

used, for transforming and analyzing data 356, 357

**API administration** 74, 75

Microsoft 365 administration 75

Power BI administration 76, 77, 78

Power Platform administration 76

**application lifecycle management (ALM)** 123, 128

Azure DevOps 124

best practices 155

environment complexity 128, 129

for Power BI 151, 152

for Power Platform 130

for solution components 155

NuGet developer tools and assemblies 123

Power Platform solution complexity 129, 130

**application lifecycle management, best practices**

multiple solutions, using 441-443

single solution, using 441

specific solution package, creating 441

unmanaged or managed solutions, using 441

**architectural best practices**

administration and

monitoring 95, 96, 438, 439, 440

development and testing environments, separating 428, 429

environment regions 94, 438

environment strategy 87, 432

multiple tenants, using 428

single tenants 83

unsupported integration topology 429, 430

**authentication** 218, 221

**authentication and authorization, Microsoft cloud ecosystem** 219, 220

authorization, granting 219, 221

licenses, assigning 219, 221

user identity, provisioning 219, 220

**authentication features, for internal users** 228

ADFS claim rules 229

conditional access 228

cross-tenant inbound restriction 231

cross-tenant outbound restriction 230

multi-factor authentication 229

single sign-on options 230

**authentication governance, for internal users**

Dataverse session governance 236

Dataverse user accounts provisioning governance 235, 236

- authorization** 218, 239  
in Power Platform 239-241
- authorization, in Dataverse and model-driven apps** 241  
access teams 253  
basic authorization, setting up 246  
column-level security 254  
fundamentals 241  
group teams 249  
hierarchy security 252  
model-driven app access, authorizing 250, 251  
Modernized Business Units 248, 249  
record sharing 253  
standard role-based security 247, 248  
user interface security 254
- automated flows** 11
- Azure Active Directory (AAD)** 34  
guest users 237  
integration 351
- Azure Active Directory B2C** 34
- Azure API Management** 35, 359  
integrating with 359
- Azure Aware plug-in** 368
- Azure Blob storage** 36  
integrating with 351
- Azure Cosmos DB** 36  
integrating with 355
- Azure Cosmos DB plug-in** 379
- Azure Data Factory** 406  
dataflows 407  
datasets 407  
pipelines and activities 406
- Azure Data Factory solution** 356
- Azure Data Lake**  
integrating with 352-354
- storage 355
- Azure DevOps** 37, 124, 125, 181  
capabilities 124, 181  
for Power Platform 143  
using, for administration 80
- Azure DevOps, with Power Platform Build Tools**  
Pipeline versus Release 150  
solutions, committing to source control 146  
solutions, distributing between development environments 146-148  
solutions, distributing out of development 148
- Azure Event Hub integration** 351
- Azure Event Hubs** 34  
integration 351
- Azure Functions** 35
- Azure IoT Central** 37, 342
- Azure IoT Hub** 37, 342
- Azure Key Vault** 37
- Azure Logic Apps** 35, 358  
integrating with 358
- Azure Monitor** 37
- Azure Service Bus** 34, 59  
integration 351
- Azure SQL** 35  
integrating with 355
- Azure Synapse Analytics** 36  
integrating with 352-354
- Azure Synapse Link for Dataverse**  
advanced integration scenarios 355
- B**
- backend components**  
testing 122

- backend integration**  
  alternate keys  
  best practices 385  
  patterns 364
- backend solution approaches** 364
- batch requests** 385
- Bring-Your-Own-Key (BYOK)** 171
- built-in plug-in/ServiceBusPlugin** 368
- Business Process Flows (BPFs)** 108
- button flows** 11
- C**
- canvas apps** 9, 383  
  and Power Automate customization 326  
  and Power Automate extensibility 326  
  app authorization 255  
  authorization 255  
  connections authorization 255
- canvas apps, sharing types**  
  co-owner sharing 255  
  user sharing 255
- capacity restrictions, Power Platform environments**  
  API limits 56  
  request limits and allocations 55  
  storage capacity limits 54, 55
- CDS Default Publisher** 135
- Center of Excellence Starter Kit**  
  reference link 95
- central roles and responsibilities**  
  customer procurement 189  
  customer project sponsor 189  
  executive steering committee 189  
  partner sales team 189  
  Project Management Office (PMO) 189, 190
- citizen developer** 104  
  versus IT pro developer 104, 105
- Clone a patch functionality** 140
- Clone solution** 141
- cloud solutions** 383
- combined model** 176
- Common Data Model (CDM)** 6, 7
- complex testing strategy** 435
- compliance** 260, 464
- Compliance Manager**  
  reference link 260
- Computer Telephony Integration (CTI)** 319
- Configuration Migration Tool** 398  
  working 399, 400
- connector groups**  
  blocked data 58  
  business data 57  
  non-business data 58
- Contoso Inc.** 4, 24  
  application lifecycle management,  
    implementing 127  
  data migration design 419- 422  
  data migration, planning 390  
  implementation, with Power Platform  
    applications 24, 25
- integration, establishing between Dynamics 365 applications and Exchange** 32
- implementation project, preparing** 167, 168
- integration, establishing between Dynamics 365 applications and Microsoft Cloud App Security** 33
- integration, establishing between Dynamics 365 applications and Microsoft Intune** 32
- integration, establishing between Dynamics 365 applications and Microsoft OneDrive** 32

- integration, establishing between Dynamics 365 applications and Microsoft OneNote 32
- integration, establishing between Dynamics 365 applications and Microsoft Outlook 32
- integration, establishing between Dynamics 365 applications and Microsoft Teams 32
- integration, establishing between Dynamics 365 applications and SharePoint 32
- Microsoft 365, using 32
- Microsoft Azure, using 38
- planned Power Platform solution, architecting 46
- Power Platform solution security, designing 217
- Power Platform integration, designing 340
- Contoso Inc. ALM strategy**
  - ALM decisions 162
  - Azure DevOps, establishing 160
  - Power BI ALM 161
  - Power Platform solutions, using 161
  - practices 160
- Contoso Inc. implementation project** 211
  - bidding process 212
  - documentation 213
  - management tools 213
  - plan 213
  - project setup and methodology 212, 213
  - setup 214, 215
- Contoso Inc. Power Platform architecture** 97
  - clients 100
  - production environment 99, 100
  - tenant structure 98
  - user groups and licensing 101
- Contoso Inc. Power Platform integration design** 386
  - backend integration 387
- frontend integration 386
- with Microsoft 365 386
- with Microsoft Azure 386
- Contoso Inc. Power Platform solution design** 276, 334
  - automations 335
  - client-side extensibility 335, 336
  - decisions 336
  - model-driven apps 334
  - server-side extensibility and integrations 336
- Contoso Inc. project team workplace setup**
  - citizen developers, enabling 126
  - core project team, enabling 125, 126
- Contoso Inc. security architecture** 271
  - Active Directory integration 271
  - Azure Active Directory (AAD) 218
  - Data Loss Prevention (DLP) policies 272
  - Dataverse 272
  - security decisions 272, 273
- Contoso Inc., with Microsoft Azure**
  - application life cycle management 38
  - authentication and single sign-on, using 38
  - Azure Monitor, establishing 38
  - IoT integration 38
  - Power Platform integration 38
- CRMRestBuilder** 121
- cross-tenant restrictions** 230
- custom backend integrations** 381
- custom connectors**
  - building 328, 329
- custom development tools**
  - CRMRestBuilder 121
  - NuGet developer tools 120
  - Postman 121
  - Power Apps Command-Line Interface (CLI) 119

- Power Platform extensions, for Visual Studio 120  
presenting 118  
testing tools 122  
Visual Studio 118  
Visual Studio Code 119  
XrmToolBox 121
- customer cloud structure, Power Platform architecture** 47, 48  
app registration 49  
group management 49  
license management 49  
Office 365 Activity Logging 49  
user management 49
- customer data platform (CDP)** 20
- customer enterprise architecture and environment** 169, 170  
authentication providers 170  
data protection requirements 171  
data residency requirements 170  
internet restrictions 170
- customer roles and responsibilities**  
Adoption and Change Management (ACM) 197  
architecture board 197  
business process owner 195  
customer program manager 195  
customer project manager 195  
development 196, 197  
enterprise architecture 197  
IT lead 196  
IT services 196  
key user/tester 195
- Customer Service Hub** 16
- custom plug-in** 366, 369, 379
- custom workflow action (CWA)** 366
- D**
- data connectors** 10, 11
- dataflows**  
Power Query 400, 401
- dataflows designer** 115  
capabilities 115
- data integration pattern** 376, 377  
standard Azure data integrations 380  
virtual tables 377
- Data Loss Prevention (DLP) policies** 57  
environment level 57  
tenant level 57  
used, for placing connectors into groups 57, 58
- data migration**  
advanced migration 392, 393  
as part of integration 390, 391  
best practices 462-472  
code, using 408  
of consolidated data 392  
overview 390
- data migration best practices**  
any type document, migrating 469  
API limits, resolving 471  
business process flow issues 467  
compliance considerations 464  
contractual responsibility, including 464, 465  
encoding issues of legacy IT systems 465  
execution time, planning 471  
lack of knowledge 464  
mapping issues 466  
migrated data quality, verifying 472  
migration automation 470  
migration steps order, following 469  
performance 470

- physical access, obtaining 464  
project duration 462  
record ownership issues, resolving 465, 466  
record status issues 468  
records with record relationship issues,  
excluding 467  
scope, determining 462, 463  
storage impact 463  
system fields, setting 468
- data migration challenges and best practices 409**
- access issues 410
  - API limits, resolving 417, 418
  - business process flow issues 413, 414
  - compliance considerations 410
  - data verification, by customer 418
  - documents, migrating 415
  - encoding issues 412
  - impact on storage 410
  - lack of documentation 411
  - lack of knowledge, copying with 411
  - mapping issues 412, 413
  - migration automation 416
  - order of migration 415, 416
  - performance 417
  - planning and effort estimation 409
  - poor-quality source data 411
  - record ownership issues 412
  - record relationship issues 413
  - record status issues 414
  - scoping 409
  - system fields, setting 415
  - time, calculating for migration  
execution 418
- data migration tools and techniques 394**
- data, entering manually 394
  - Excel files, using 394-396
- data protection 260**
- Data Provider Plug-ins 378**
- Dataverse**
- content, embedding into third-party  
containers 362
  - third-party content, embedding  
into 360, 362
- Dataverse applications**
- implicit integrations 341
- Dataverse auditing 82**
- Dataverse authentication 231**
- OAuth authentication 232-234
  - Office 365 authentication 232
- Dataverse authorization**
- business units 242-244
  - record ownership 244
  - security roles and permissions 244-246
  - teams, types 242
  - users and teams 241
- Dataverse automation 292**
- business process flows 296, 297
  - business rules 292, 293
  - Classic Dataverse workflows 293, 294
  - custom actions 294
  - custom APIs 295
- Dataverse-based application**
- layer security roles, using 446, 447
- Dataverse client-side extensibility 297**
- canvas apps, embedding 306, 307
  - Power Apps Component  
Framework 299, 300
  - standard custom controls 297-299
  - web resources 301
- Dataverse connectors**
- Dataverse (legacy) 322
- Dataverse content-based security 263, 448**
- business unit hierarchy, using 448
  - business units, using 264

- client-side scripting or business rules, using 264
- server-side event handlers, using 264
- table form switching, using 264, 449
- Dataverse data import wizard**
  - using 397, 398
- Dataverse data modeling**
  - customer-specific standard customization 281
  - Dynamics 365 application 280
  - third-party Dataverse-based application 281
- Dataverse security roles**
  - layering 262, 263
  - modifying 262
- Dataverse server-side extensibility 308, 457**
  - and automation options 459, 460
  - API interface types 308, 309, 310
  - Azure Event Hub integration 316
  - Azure Service Bus integration 313-316
  - custom workflow actions 313
  - Dataverse API, using 458
  - external applications, building 317
  - plug-in event handlers 311, 312
  - synchronous workflows and plug-ins, avoiding 460
  - webhook integration 316
  - webhook integration 317
- Dataverse standard customization 278-280**
  - automations 280
  - data modeling 280
  - Dataverse data modeling 281
  - user interface design 280, 284
- Dataverse user interface design 284**
  - customization artifacts 290
  - table charts 287, 288
  - table forms 284, 285
- table-specific and table-independent dashboards 288-290
- table views 285-287
- dedicated environment 90**
  - complex solution/complex team structures 90, 434
- dependencies and solution segmentation**
  - layering behavior 138, 139
  - solution dependencies 139
  - solution layering 137, 138
  - solution segmentation 139
- desktop clients**
  - browser client 62
  - Dynamics 365 App for Outlook 63
  - Omnichannel for Dynamics 365 Customer Service 65
  - Power Automate Desktop flows 65
  - Unified Service Desk 63
- desktop flows 11, 323**
  - attended mode 325
  - manual creation 324
  - recording 324
  - unattended mode 325
- dual-write 380**
  - reference link 380
- Dynamics 365**
  - reference link 23
- Dynamics 365 App for Outlook**
  - capabilities 63
  - features 63
- Dynamics 365 clients 62**
  - desktop clients 62
  - mobile clients 66
- Dynamics 365 Connected Field Service 342**
- Dynamics 365 Marketing 342**
- Dynamics 365 Sales 341**

**E**

- Email Server Profile** 343
- encryption at rest** 261
- environment strategy** 87
  - complex testing 91
  - dedicated environment 90
  - default environment 88
  - developer environment 88
  - developer sandbox environment 94
  - feature testing environment 94
  - multiple release strategy 91, 92
  - other types 94
  - product upgrades 92, 93
  - shared test and production environment 88, 89
  - training environment 94
- environment strategy, for enterprise-scale project** 432, 433
  - complex solution/complex team structures 434
  - complex testing strategy 435
  - dedicated environments strategy 434
  - multiple release strategy 436
  - product upgrades strategy 437
  - shared test and production environment strategy 433
- event-driven integration** 363
- Excel Online** 31
- Exchange**
  - Dataverse, integrating with 342, 344
- Exchange Online** 342
- Exchange on-premises** 342
- execution phase** 203
  - final testing 208, 209
  - initial analysis 204, 205

iterative execution 205  
project initiation 204  
project preparation 204  
solution deployment 209

**extensibility**

configuration 277  
custom development 277  
customization 277  
overview 276  
standard capabilities 277

**extensibility best practices** 456

client-side extensibility performance, optimizing 456, 457

Dataverse server-side extensibility 457

**external applications, building**

alternate clients and portals 318  
batch jobs 317  
mobile clients 318

**external users authentication** 237-239

external 237  
invitation codes 239  
local 237  
open registration 239  
options, in Power Pages configuration area 237

**extract-transform-load (ETL)** 377, 392**F****feasibility**

economic feasibility 199  
legal feasibility 199  
operational feasibility 199  
scheduling feasibility 199  
study 199

**first-party Power Apps** 15**frontend integration**

- 
- best practices 384
  - patterns 360
  - frontend solution approaches** 360
  - G**
    - general availability (GA)** 93, 437
    - General Data Protection Regulation (GDPR)** 261
    - GitHub**
      - for Power Platform 150, 151
    - guest users** 255
  - H**
    - hackathon** 202
    - hierarchy security**
      - manager hierarchy 252
      - position hierarchy 252
    - Hosted Application Toolkit (HAT)** 364
    - hotfixes** 143
  - I**
    - identity and access management (IAM)** 452
    - identity and authentication solutions, for internal users** 221
      - cloud identity approach 222, 227
      - federation approach 226-228
      - pass-through authentication approach 224, 225
      - password hash synchronization approach 223, 224
  - IFrame configuration**
    - custom method 360
    - standard method 360
  - inactivity timeout** 236
  - independent software vendor (ISV)** 118
  - Information Technology Infrastructure Library (ITIL)** 172
  - Infrastructure as a Service (IaaS)** 33
  - initial data load approach** 390
  - integration best practices** 460
    - backend integration 461
    - frontend integration 461
  - ISV Studio** 118
    - reference link 118
  - iterations** 175
  - iterative executions** 205
    - iterative analysis 206
    - iterative design 206
    - iterative development 207
    - iterative testing 208
  - iterative model** 175, 176
  - IT pro developer** 104
    - versus citizen developer 104, 105
  - IT security**
    - authentication, versus authorization 218
    - overview 218
  - J**
    - jQuery** 363
  - K**
    - kick-off meeting** 204
  - L**
    - Life Cycle Services (LCS)** 380
    - listener** 313
    - low-code/no-code approach** 326

**M****managed environments** 60

data policies 60

features 60

sharing limits 60

weekly digest 60

**managed properties** 136

columns properties 136

for artifact types 136

tables properties 136

**managed solutions** 441**mass integration** 376**Microsoft 365** 28

integrating with 341

Microsoft Enterprise Mobility + Security 31

Microsoft Office 365 29

product groups 29

**Microsoft 365 Admin Center**

monitoring analytics 68

reference link 68

tasks 68

**Microsoft 365 services**

integrating with 342

**Microsoft AppSource** 118

reference link 118

**Microsoft Authentication Library (MSAL)** 318**Microsoft Azure** 33, 34

Azure Active Directory 34

Azure API Management 35

Azure Event Hubs 34

Azure Logic Apps 35

Azure Service Bus 34

integrating with 341

**Microsoft Azure licensing** 39

Microsoft 365 Business 39

Microsoft 365 Education 39

Microsoft 365 Enterprise 39

overview 39

**Microsoft Azure services**

integrating with 351

**Microsoft Cloud App Security** 31**Microsoft cloud ecosystem**

authentication and authorization 219

**Microsoft Dataverse** 6, 7, 105

components 7

storage types 53

model-driven app designer, capabilities 108

model-driven app designer, sitemap 108

Power Apps Maker Portal 106, 107

storage types 53

**Microsoft Dataverse for Teams** 53**Microsoft Dynamics 365**

implicit integrations 341

**Microsoft Dynamics 365 Business Central** 19

capabilities 19

**Microsoft Dynamics 365 Commerce** 18

capabilities 19

**Microsoft Dynamics 365 Connected Spaces** 21**Microsoft Dynamics 365 CRM applications** 15**Microsoft Dynamics 365 Customer Insights** 20

capabilities 20

**Microsoft Dynamics 365 Customer Service** 16

capabilities 16

**Microsoft Dynamics 365 ERP applications** 17**Microsoft Dynamics 365 Field Service** 16

capabilities 16

**Microsoft Dynamics 365 Finance** 17

capabilities 18

- Microsoft Dynamics 365 Fraud Protection** 21
- Microsoft Dynamics 365 Guides** 21
- Microsoft Dynamics 365**
- Human Resources**
  - capabilities 19
- Microsoft Dynamics 365 Marketing** 15
- capabilities 16
- Microsoft Dynamics 365 Product Visualize** 21
- Microsoft Dynamics 365 Project Operations** 17
- capabilities 17
- Microsoft Dynamics 365 Remote Assist** 21
- Microsoft Dynamics 365 Sales** 15
- capabilities 15
- Microsoft Dynamics 365 Sales Insights** 20
- Microsoft Dynamics 365 Supply Chain Management**
- capabilities 18
- Microsoft Dynamics 365 Unified Service Desk** 22
- Microsoft Enterprise Mobility + Security** 31
- Microsoft Cloud App Security 31
  - Microsoft Intune 31
  - System Center Configuration Manager 32
- Microsoft Excel** 31
- Microsoft Exchange** 30
- Microsoft Office 365** 29
- Microsoft Excel 31
  - Microsoft Exchange 30
  - Microsoft OneDrive 30
  - Microsoft OneNote 30
  - Microsoft Outlook 30
  - Microsoft SharePoint 30
  - Microsoft Teams 30
  - Microsoft Word 31
- Microsoft OneDrive** 30
- Microsoft OneNote** 30, 348
- integrating with 348, 349
- Microsoft Outlook** 31
- Microsoft Power Platform** 4-6
- Microsoft Power Platform licensing**
- overview 22-24
- Microsoft Privacy Statement**
- reference link 260
- Microsoft Project desktop client** 180
- Microsoft Project for Web** 181
- Microsoft Project Online** 180
- Microsoft Service Trust Portal**
- reference link 260
- Microsoft SharePoint** 30
- Microsoft SQL Server Integration Services (SSIS)** 402
- data, extracting 402-405
  - data, loading 406
  - data, transforming 405
- Microsoft Teams** 30
- Microsoft Teams integration**
- capabilities 346, 347
  - configuring, steps 347
- Microsoft Trust Center**
- reference link 260
- Microsoft updates** 142
- first-party applications updates 142
  - Power Platform hotfixes 143
  - Power Platform updates 142
- Microsoft Word** 31
- mobile application management (MAM)** 31
- mobile apps**
- designing 292
- mobile clients** 66
- Dynamics 365 for Phones and Dynamics 365

for Tablets 66  
Field Service Mobile 66  
Power Apps mobiles 66  
Power Automate mobile 66  
Power BI app 66  
**mobile device management (MDM)** 31  
**model-driven applications** 7-9, 383  
and Dataverse relationship 279  
capabilities 7  
components 8  
designing 291  
**Monitor tool** 110  
**multiple release strategy** 436  
**multiple solutions**  
component sharing and component libraries 158  
using 156-158  
**multi-tenant environments**  
development and testing 83, 84  
unsupported integration topology 84

## N

**network traffic analyzers**  
using 122, 123  
**Non-Disclosure Agreement (NDA)** 202  
**NuGet developer tools and assemblies** 120  
code generation tool 120  
configuration migration tool 123  
package deployer tool 123  
plug-in registration tool 121  
solution packager tool 124

## O

**Office Add-In** 63  
**Omnichannel for Dynamics 365 Customer Service** 65  
**on-demand frontend integration** 363  
**OneDrive**  
integrating with 345, 346  
versus SharePoint 345  
**OneNote integration process**  
configuration steps 349  
**On-Premises Data Gateway** 13, 58, 328  
components 59  
types 58  
**Open Data Initiative (ODI)** 6  
**operational acceptance testing (OAT)** 91  
**operation phase** 209, 210  
decommission 211  
maintenance 210  
support 211  
support transition 210

## P

**paginated reports** 117  
**partner roles and responsibilities** 190  
business consultant 192  
data migration lead 194  
developers 193  
development lead 193  
infrastructure consultant 194  
integration lead 194  
lead architect 191  
program manager 190  
project manager 191

- release manager 194
- solution architect 192
- technical consultant 193
- test team 194
- Platform as a Service (PaaS) 33**
- platform auditing**
  - Dataverse auditing 82
  - Office 365 Activity Logging 81
- Portals Web API 320**
- Postman 121**
  - capabilities 121
  - reference link 121
- Power Apps 383**
- Power Apps Command-Line Interface (CLI) 119**
- Power Apps Component Framework (PCF) 299**
- Power Apps Maker Portal 106, 107**
  - reference link 106
- Power Apps Studio 109**
  - capabilities 110
- Power Automate 11, 382**
  - authorization 256
  - background flows authorization 256
  - interactive flows authorization 257
  - using, for administration and monitoring 78, 80
- Power Automate Desktop flows 257**
  - using, for robotic process automation 65
- Power Automate flows**
  - cloud flows 321, 322
  - desktop flows 323-325
  - presenting 321
  - Process advisor 325
- Power Automate Maker Portal 112**
  - capabilities 112
  - reference link 112
- Power BI 12**
  - application lifecycle management 151-154
  - authorization 257
  - best practices 445
  - capabilities 12, 13
  - capacity 61
  - components 153
  - concepts 13
  - dashboards 62
  - dataflows 62
  - datasets 61
  - deployment pipelines 154
  - environments 152
  - integrating, with Dataverse 267
  - practices 160
  - reports 62
  - structure 60
  - used, for analyzing 357
  - workbooks 62
  - workspaces 61
- Power BI admin portal**
  - administration capabilities 67
  - capabilities 70
  - reference link 67, 70
- Power BI authentication 235**
- Power BI authorization**
  - DirectQuery 257
  - Import 257
- Power BI Builder 117**
  - capabilities 117
- Power BI components**
  - Power BI file (PBIX) 153
  - Power BI template (PBIT) 153

- Power BI dataflows** 383
- Power BI deployment pipelines** 154
- Power BI designer tools** 115
- Power BI Desktop 116
  - Power BI service 117
- Power BI Desktop** 116
- capabilities 116
- Power BI extensibility**
- presenting 329, 330
  - presenting, best practices 330
- Power BI extensibility best practices**
- client-side interface extensibility 330, 331
  - server-side extensibility 331
- Power BI integration** 383
- Power BI service**
- capabilities 117
  - reference link 117
  - using 117
- Power Fx** 9, 326, 327
- Power Pages** 14, 342
- authorization 258, 259
  - capabilities 14
- Power Pages extensibility**
- presenting 320
- Power Pages portal authentication structure**
- contact 259
  - Table Permission 259
  - Web Page Access Control Rules 259
  - web role 259
- Power Pages portals/public website** 383
- Power Pages Studio** 111, 112
- Power Platform**
- administration and monitoring 67
  - authorization 239, 240, 241
  - environments 50
- reference link 23
- technology 50
- Power Platform actions, capabilities**
- administration tasks 150
  - helper tasks 150
- Power Platform Admin Center** 68
- administration and monitoring capabilities 69
  - capacity analytics 69
  - Dataverse analytics 70
  - Power Apps analytics 70
  - Power Automate analytics 70
- Power Platform administration and monitoring**
- administration centers 67
  - API administration 74
  - application monitoring 82
  - Azure DevOps, using 80
  - platform auditing 80
  - Power Automate, using 78
  - PowerShell administration and monitoring 71
- Power Platform administration centers** 67
- Microsoft 365 Admin Center 68
  - Microsoft Azure portal 67
  - Power BI admin portal 70
  - Power Platform Admin Center 68
- Power Platform architecture** 46
- customer cloud structure 47, 48
  - Microsoft cloud infrastructure 46
- Power Platform Build tools** 125
- environment tasks 145
  - helper and quality check tasks 144
  - overview 143
  - solution tasks 144
- Power Platform Data Connectors**
- Actions capability 57

- custom connectors 56
- premium connectors 56
- standard connectors 56
- Tables capability 57
- Triggers capability 57
- Power Platform environments** 51, 52
  - capacity restrictions 53
  - Data Loss Prevention policies 57
  - default 51
  - developer 51
  - main components 52
  - managed environments 60
  - Microsoft Dataverse 53
  - Microsoft Dataverse for Teams 53
  - Microsoft Teams 51
  - On-Premises Data Gateway 58, 59
  - Power Platform Data Connectors 56, 57
  - production 51
  - Sandbox 51
  - support 51
  - trial 51
- Power Platform implementation approach**
  - overview 168
  - preparing 104
  - recommendations 168
- Power Platform integration** 381
  - need for 340, 341
  - overview 340, 341
- Power Platform integration, best practices** 384
  - backend and frontend integration 384
- Power Query**
  - using, with Dataflows 400, 401
- PowerShell administration and monitoring** 71
  - Microsoft 365 administration 71
  - Power BI administration 73
- Power Platform administration 72, 73
- PowerShell monitoring 74
- Power Virtual Agent** 381, 382
  - capabilities 12
- Power Virtual Agents (PVA) designer**
  - capabilities 113
  - reference link 113
- preparation phase** 198
  - approval, seeking 200
  - budget, specifying 200
  - contract 203
  - demand, identifying 198
  - discovery 203
  - feasibility study 199
  - negotiations 203
  - Request for Information (RFI), issuing 200
  - RFP/RFQ/RFT, issuing 201
- privacy** 260
- product upgrades strategy** 437
- project documentation**
  - creating 182
  - project plan 182, 183
  - requirements document 184
  - solution architecture document 184
  - solution/technical design document 185
  - testing 185
  - training 186
- project effort estimation**
  - business requirements 177
  - custom development 178
  - data migration 179
  - efforts 180
  - infrastructure requirements 178
  - integration 178, 179
  - making 177

- project implementation**  
methodologies 172, 173  
methodologies and tools 171  
programs and projects 171
- project implementation methodologies** 172  
agile model 172-175  
approaches 172, 173  
combined model 176  
iterative model 172-176  
waterfall model 172-174
- project management tools** 180  
Azure DevOps 181  
effort estimators 181  
Microsoft Project 180
- Project Manager (PjMgrs)** 215
- project phases** 198  
execution phase 203  
operation phase 209  
preparation phase 198
- project roles and responsibilities** 188, 189  
central roles and responsibilities 189  
customer roles and responsibilities 194  
partner roles and responsibilities 190
- project types** 186  
external project 187  
internal project 186, 187
- Proof of Concept (POC)** 201
- publish-subscribe pattern** 370-374
- R**
- relay pattern** 367, 368
- remote procedure call (RPC) pattern** 364
- reports** 12, 115-117, 362
- request-callback pattern** 374-376
- Request for Information (RFI)**  
issuing 200
- RetrieveMultiple Dataverse events** 449
- reusable components** 444
- Ribbon Workbench** 109, 305  
command bar 109
- robotic process automation (RPA)** 323
- row-level security (RLS)** 258
- S**
- Sales Hub** 15
- scale groups** 50
- scheduled flows** 11
- scope creep** 173
- seamless single sign-on** 224, 230
- security best practices** 261, 446  
custom roles, creating 446  
Dataverse content-based security 263, 448  
Dataverse security roles 262, 446  
identity and access management  
automation, using 267, 268, 453, 454  
Power Platform mature security model,  
establishing 269-271, 454, 455  
security across solution components,  
integrating 450, 451  
security, integrating across solution  
components 265, 266
- security integration**  
Dataverse-Power BI integrated security 452  
Dataverse-SharePoint integrated  
security 451
- segmentation**  
best practices 445  
using 444

- server-side extensibility best practices**
  - automation options 332, 333
  - Dataverse API selection 331
  - performance impact 334
- service authentication, for internal users** 231
  - Dataverse authentication 231
  - Power BI authentication 235
- session timeout** 236
- shared test and production environment strategy** 433
- SharePoint**
  - Dataverse, integrating with 344, 345
  - integrating, with Dataverse 266
  - versus OneDrive 345
- Show dependencies function** 139
- single publisher**
  - using 445
- single solution**
  - using 156
- site map** 8, 108
- Sitemap Designer** 109
- Skype for Business**
  - Dynamics 365, integrating with 349, 350
- soft booking** 203
- Software as a Service (SaaS)** 37, 50
- solution best practices**
  - general practices 155
  - segmentation, using 158, 159
  - source control, using 159
  - unmanaged, versus managed 156
- Solution Layers** 139
- solution management**
  - dependencies and solution segmentation 137
  - managed properties 136
- solution patches** 140
- solution properties** 133
  - configuration pages 134
  - solution publisher 133
  - solution version 134
- solution publisher**
  - best practices 159
  - Choice Prefix 133
  - prefix 133
- solutions**
  - environment variables 132
  - overview 131, 132
  - properties 133
  - structuring 156
- solutions management** 131
- solution types** 134
  - Common Data Services Default Solution 135
  - default solution 135
  - managed solution 135
  - unmanaged solution 134
- solution updates** 140, 142
- source control**
  - using 445
- SQL Server Reporting Services (SSRS)** 449
- Staging database** 393
- standard OData 4.0 plug-in** 378
- Statement Of Work (SOW)** 202
- stub users** 465
- supporting organization levels**
  - first-level 211
  - second-level 211
  - third-level 211
- support transition services** 210
- synchronous plug-ins** 334, 460

**System Center Configuration Manager** 32  
**system integration testing (SIT)** 91  
**Systems Development Life Cycle (SDLC)** 172

## T

**table artifacts**  
charts 279  
dashboards 280  
forms 279  
views 279  
**table forms** 362  
**table form types, Dataverse user interface design**  
Card Form 284  
Main Form 284  
Quick Create Form 284  
Quick View Form 284  
**table-independent artifacts**  
custom pages 280  
dashboards 280  
table-independent artifacts 280  
**table views** 362  
**table views, Dataverse user interface design** 285  
calendar views 286  
editable views 286  
personal views 286  
public views 286  
system views 286  
**Tabular Data Stream (TDS) endpoint** 231, 462  
**testers** 194  
**test leads** 194  
**third-party containers and content**  
Dataverse content, embedding into 360-362

**Transparent Data Encryption (TDE)** 261  
**triggers** 322

## U

**Unicode data encoding** 412, 465  
**Unified Service Desk extensibility**  
USD client 319  
USD packages 319  
**Unified Service Desk (USD)** 63  
Agent Desktop application 64  
Dataverse solution 64  
using 364  
**unit testing** 193  
**unmanaged solutions** 134, 441  
**unsupported integration topology** 84, 85, 430  
central consolidation  
environment 86, 430, 431  
central reporting environment 86, 87, 431  
URL 429  
**UPsert request** 385  
**User Acceptance Testing (UAT)** 208  
**user interface**  
testing 122  
**User Interface Integration (UII)** 64

## V

**Virtual connectors, Dataverse** 379  
**virtual tables** 282, 283, 377, 378  
**Visual Studio** 118  
Power Platform extensions 120  
**Visual Studio Code** 119

## W

**waterfall model** 173

**WebHook integration** 366

**web resources**

code 302

command bar/ribbon extensibility 304, 305

data 301

form scripting 302, 303

graphical 301

overview 301

usage scenarios 301

user interface 301

web components, embedding 302

**Word Online** 31

## X

**XrmToolBox** 108, 121

reference link 108

Ribbon Workbench 109

**XRM Tooling assemblies** 332, 458

# Download a free PDF copy of this book

Thanks for purchasing this book!

If you'd like to read on the go, or your eBook purchase is not compatible with your device, you can now get a DRM-free PDF version with any Packt book at no extra cost.

To do so, follow these simple steps:

1. Scan the QR code or visit the link below



<https://packt.link/free-ebook/9781804612637>

2. Submit your proof of purchase
3. That's it! We'll send your free PDF and other benefits to your email directly



