

Microsoft Power Platform Enterprise Architecture

A guide for architects and decision makers to craft complex solutions tailored to meet business needs



Robert Rybaric



Microsoft Power Platform Enterprise Architecture

A guide for architects and decision makers to craft complex solutions tailored to meet business needs

Robert Rybaric

Packt

BIRMINGHAM—MUMBAI

Microsoft Power Platform Enterprise Architecture

Copyright © 2020 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.

Commissioning Editor: Richa Tripathi

Acquisition Editor: Alok Dhuri

Senior Editor: Rohit Singh

Content Development Editor: Nithya Sadanandan

Technical Editor: Pradeep Sahu

Copy Editor: Safis Editing

Project Coordinator: Deeksha Thakkar

Proofreader: Safis Editing

Indexer: Priyanka Dhadke

Production Designer: Vijay Kamble

First Published: September 2020

Production reference: 1240920

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

B3 2PB, UK.

ISBN 978-1-80020-457-7

www.packtpub.com



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

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at packt.com and, as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at customercare@packtpub.com for more details.

At 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.

Contributors

About the author

Robert Rybaric is a Microsoft Power Platform and Microsoft Dynamics 365 architect and certified trainer. He also holds several other certifications in Microsoft 365, Microsoft Azure, Microsoft SQL Server, and Microsoft BizTalk Server, TOGAF 9, ITIL 2011, IBM, and Oracle.

Robert has worked as an architect for the Microsoft Corporation on numerous Microsoft Dynamics 365 presale and project implementation activities for enterprise customers across Europe.

He is now a freelance architect and trainer, implementing Microsoft Dynamics 365 solutions for global customers, and leads many Microsoft Power Platform training sessions.

About the reviewers

Jürgen Leitz is a solution architect/consultant who has worked with Microsoft Dynamics for more than 15 years. He has worked with Microsoft Dynamics since the very first version of Microsoft Dynamics CRM, right through to today and Microsoft Dynamics 365 and Power Platform, mostly working in projects for large enterprise customers.

Initially, he worked as a Microsoft Partner. For the last 10 years, he has been working for Microsoft, acquiring expertise in Microsoft Dynamics 365 solution architecture, as well as in functional and technical aspects of the Dynamics/Power Platform product area, and also in related technology components in Microsoft Azure.

Miklos Hoffmann has 30 years of IT experience. He has spent the last 20 years in the world of customer relationship management with a number of different technologies, including Siebel, SAP, and Microsoft CRM, beginning with version 1.2.

He has worked on enterprise Dynamics 365 implementations in various European countries in the manufacturing, finance, and retail industries. His particular interests are cloud migration and integration, and he participated as an architect in the cloud migration of a 12,000+ user system based on Microsoft Dynamics 365 Customer Engagement.

"Technology is important; however, the main question is whether the project addresses genuine business needs and how the team can sit in the same boat from the customer and supplier sides to reach the desired destination."

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit 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.

Table of Contents

Preface

Section 1: The Basics

1

Microsoft Power Platform and Microsoft Dynamics 365 Overview

Introducing Contoso Inc.	20	Microsoft Dynamics 365 Customer Service	31
Introducing Microsoft Power Platform	20	Microsoft Dynamics 365 Field Service	31
Introducing the Common Data Model and Common Data Service	23	Microsoft Dynamics 365 Project Operations	32
Introducing model-driven apps	24	Introducing Microsoft Dynamics 365 ERP applications	32
Introducing canvas apps	25	Microsoft Dynamics 365 Finance	33
Introducing Power Automate	26	Microsoft Dynamics 365 Supply Chain	33
Introducing Power Virtual Agents	27	Management	33
Introducing Power BI	28	Microsoft Dynamics 365 Commerce	34
Introducing On-Premises Data Gateway	28	Microsoft Dynamics 365 Human Resources	34
Introducing AI Builder	28	Microsoft Dynamics 365 Business Central	34
Introducing Power Apps portals	29	Introducing Microsoft Dynamics 365 AI, MR, and other modules	35
Introducing Dynamics 365 Customer Voice	29	Microsoft Dynamics 365 Customer Insights	35
Introducing Microsoft Dynamics 365 CRM applications	30		
Microsoft Dynamics 365 Sales	30		
Microsoft Dynamics 365 Marketing	30		

Microsoft Dynamics 365 Sales Insights	36	Visualize	38
Microsoft Dynamics 365 Customer Service Insights	36	Microsoft Dynamics 365 Import Tool	38
Microsoft Dynamics 365 Product Insights	37	Microsoft Dynamics 365 Unified Service Desk	38
Microsoft Dynamics 365 Connected Store	37	Microsoft Power Platform licensing overview	38
Microsoft Dynamics 365 Fraud Protection	37	Simple relationship management	41
Microsoft Dynamics 365 Remote Assist	37	Complex marketing management	42
Microsoft Dynamics 365 Guides	38	Contoso Inc. Power Platform commitment	42
Microsoft Dynamics 365 Product		Summary	44

2

Microsoft 365 and Microsoft Azure Overview

Contoso Inc. cloud maturity	46	Introducing Azure Blob Storage	54
Introducing Microsoft 365	46	Introducing Azure Data Lake Storage Gen2	54
Introducing Microsoft Office 365	46	Introducing Azure IoT Hub and Azure IoT Central	54
Overviewing Microsoft Enterprise Mobility + Security	48	Introducing Azure Key Vault	54
Contoso Inc. using Microsoft 365	49	Introducing Azure DevOps	55
Introducing Microsoft Azure	51	Introducing Azure Monitor	55
Introducing Azure Active Directory	51	Contoso Inc. using Microsoft Azure	55
Introducing Azure Service Bus	51	Microsoft 365 and Microsoft Azure Licensing overview	56
Introducing Azure Event Hub	52	Understanding Microsoft 365 licensing	56
Introducing Azure Logic Apps	52	Learning about Microsoft Azure licensing	57
Introducing Azure API Management	52		
Introducing Azure Functions	52		
Introducing Azure SQL	53		
Introducing Azure Cosmos DB	53	Summary	58

Section 2: The Architecture

3

Understanding Microsoft's Power Platform Architecture

Contoso Inc. starts architecting their planned Power Platform solution	62	Learning about the Power BI admin center	84
Understanding the Power Platform architecture	62	Understanding PowerShell administration and monitoring	85
Learning about the Microsoft cloud infrastructure	63	Learning about API administration	89
Understanding the customer cloud structure	64	Administration and monitoring using Power Automate	92
Learning about Power Platform technology	66	Administration using Azure DevOps	94
Understanding Power Platform environments	67	Learning about platform auditing	94
Learning about Power BI's structure	75	Understanding application monitoring	96
Understanding the Power Platform clients	76	Presenting architectural best practices	97
Learning about desktop clients	76	Introducing single tenants or multiple tenants	97
Understanding mobile clients	80	Understanding environment setup	101
Learning Power Platform administration and monitoring	81	Environment regions	109
Understanding Power Platform administration centers	81	Administration and monitoring	110
Contoso Inc. Power Platform architecture		Contoso Inc. Power Platform architecture	111
Tenant structure		Tenant structure	112
Power Platform environments		Power Platform environments	113
Power Platform clients		Power Platform clients	114
User groups and licensing		User groups and licensing	115
Summary		Summary	116

4

Tools and Techniques

Contoso Inc. empowering the project team	118	Introducing tools and techniques	118
--	-----	----------------------------------	-----

Introducing the citizen developer	118	development tools	134
Introducing the IT pro developer	118	Learning about Visual Studio	134
Distinguishing between the citizen developer and the IT pro developer	119	Learning about Visual Studio Code	135
Presenting configuration and customization tools	119	Introducing the Power Apps CLI	135
Knowing Common Data Service (CDS) and model-driven apps tools	119	Introducing Power Platform extensions for Visual Studio	136
Introducing PowerApps Portal Studio	124	Learning about NuGet developer tools and assemblies	136
Introducing canvas apps Designer Studio	125	Introducing XrmToolBox	137
Introducing the Power Automate designer	126	Introducing Postman	137
Introducing the Power Virtual Agents designer	128	Introducing CRM REST Builder	137
Introducing AI Builder	129	Introducing testing tools	138
Introducing the dataflows designer	130	Presenting application lifecycle management tools	139
Getting to know Power BI designer tools	130	Introducing NuGet developer tools and assemblies	139
Understanding Dynamics 365	128	Introducing Azure DevOps	140
Customer Voice designer	133	Contoso Inc. project team workplace setup	142
Understanding Microsoft AppSource	133	Enabling the core project team	142
Understanding ISV Studio	134	Enabling citizen developers	143
Presenting custom		Summary	143

5

Application Lifecycle Management

Contoso Inc. implementing application lifecycle management	146	Introducing solutions management	149
Understanding application lifecycle management	146	Getting an overview of solutions	149
Getting to know environment complexity	146	Knowing about solution types	152
Understanding Power Platform solution complexity	146	Learning about managed properties	153
ALM for the Power Platform	147	Understanding dependencies and solution segmentation	154
	148	Patching and updating solutions	157
		Learning about Microsoft updates	160

Introducing Azure DevOps for Power Platform	161	management for other solution components	171
Using Azure DevOps with Power Platform build tools	163	Application lifecycle management best practices	171
		Solution best practices	171
		Solution publisher best practices	175
		Power BI best practices	176
Application lifecycle management for Power BI	168	Contoso Inc. ALM strategy	176
Understanding environments in Power BI	168	Establishing Azure DevOps	176
Learning about Power BI components	169	Using Power Platform solutions	177
Learning about the Power BI ALM approach	169	Using Power BI ALM	178
Learning about Power BI deployment pipelines	170	Other ALM decisions	178
Application lifecycle		Summary	178

Section 3: Implementation

6

Implementation Approach and Methodologies

Contoso Inc. preparing the implementation project	182	Understanding project implementation methodologies	186
Getting an overview of the implementation approach	182	Making a project effort estimation	192
Understanding customer enterprise architecture and the environment	183	Learning about project management tools	195
Data residency example	184	Creating project documentation	197
Authentication provider example	185		
Internet access example	185	Learning about the project setup	201
Data protection example	185	Knowing project types	201
		Understanding project roles and responsibilities	202
Learning about project implementation methodologies and tools	186	Understanding project phases	213
Understanding programs and projects	186	Knowing the preparation phase	213
		Project execution phase	219
		Operation phase	225

Contoso Inc. starting the implementation project	228	Project plan, tools, and documentation	229
Bidding process	228	Project setup	230
Project setup and methodology	228	Summary	232

7

Microsoft Power Platform Security

Contoso Inc. designing Power Platform solution security	234	Understanding authorization in Power Automate	269
Getting an overview of IT security	234	Understanding authorization in Power BI	270
Authentication versus authorization	234	Understanding authorization in Power Apps portals	271
Understanding Microsoft cloud authentication and authorization fundamentals	235	Understanding compliance, privacy, and data protection	273
Understanding authentication	237	Presenting security best practices	274
Understanding identity and authentication solutions for internal users	238	CDS security roles	274
Getting to know authentication features for internal users	244	CDS content-based security	276
Learning about service authentication for internal users	247	Learning to integrate security across solution components	278
Studying authentication governance for internal users	251	Using identity and access management automation	281
Azure Active Directory guest users	253	Establishing the Power Platform mature security model	282
Authenticating external users	253	Contoso Inc. security architecture	285
Understanding authorization	256	Active Directory integration	285
Authorization in Power Platform	256	Data Loss Prevention policies	285
Authorization in CDS and model-driven apps	257	Common Data Service	285
Understanding authorization in Canvas Apps	268	Other security decisions	286
		Summary	286

8

Microsoft Power Platform Extensibility

Contoso Inc. designing the Power Platform solution	290	Building custom connectors	334
Getting an overview of extensibility	290	Presenting Power BI extensibility	335
Presenting CDS and model-driven apps extensibility	292	Knowing Power Platform extensibility best practices	336
Understanding CDS standard customization	293	CDS client-side interface extensibility	336
Understanding CDS automation	303	CDS server-side extensibility	338
Understanding CDS client-side extensibility	309	Contoso Inc. Power Platform solution design	341
Understanding CDS server-side extensibility	320	Model-driven apps	341
Power Apps Portals extensibility	331	Automations	342
Unified Service Desk extensibility	332	Client-side extensibility	342
Presenting canvas apps and Power Automate extensibility	333	Server-side extensibility and integrations	343
Introducing Canvas apps and Power Automate customization	333	Other design decisions	343
		Summary	343
		Further reading	344

9

Microsoft Power Platform Integration

Contoso Inc. designing the Power Platform integration	346	Presenting frontend integration patterns and solution approaches	363
Getting an overview of Power Platform integration	346	Embedding third-party content into CDS	363
Integrating with Microsoft 365 and Microsoft Azure	347	Embedding CDS content into third-party containers	365
Introducing implicit Dynamics 365 integrations	347	Event-driven and on-demand frontend integration	366
Learning about integrations with Microsoft 365	348	Using UI flows and Unified Service Desk	367
Learning about integrations with Microsoft Azure	356		

Presenting backend integration patterns and solution approaches	368	Learning Power Platform integration best practices	385
Remote procedure call pattern	368	Frontend integration	385
Relay pattern	371	Backend integration	386
Publish-subscribe pattern	374		
Request-callback pattern	377	Contoso Inc. Power Platform integration design	387
Data integration	379	Integration with Microsoft 365 and Microsoft Azure	388
Custom backend integrations	382	Frontend integration	388
Presenting other Power Platform integrations	383	Backend integration	389
Power Virtual Agent and AI Builder	383	Summary	390
Power BI	384		

10

Microsoft Power Platform Data Migration

Contoso Inc. planning the data migration	392	Planning and effort estimation	407
		Scoping the migration	407
Getting an overview of data migration	392	Understanding the impact on storage	408
		Compliance considerations	408
Learning about migration as part of integration	392	Understanding access issues	408
Migrating consolidated data	394	Coping with a lack of knowledge	409
Advanced migration	394	Dealing with a lack of documentation	409
Understanding data migration tools and techniques	396	Poor quality source data	409
		Understanding encoding issues	410
Doing manual entry	396	Understanding record ownership issues	410
Using Excel files	396	Understanding mapping issues	411
Using the data import wizard	398	Understanding record relationship issues	411
Introducing the Configuration Migration Tool	400	Understanding business process flows issues	412
Dataflows with Power Query	401	Understanding record status issues	412
Using SQL Server Integration Services	403	Setting certain system fields	413
Data migration challenges and best practices	407	Migrating documents	413
		Understanding the importance of the order of migration steps	413

Understanding migration automation	414	Verifying the data by the customer	416
Understanding migration performance	415	Contoso Inc. data migration design	416
Resolving API limits	415	Summary	420
Time for migration execution	416		

Other Books You May Enjoy

Index

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. You can use Power Apps to build business applications for PC and mobile devices, as well as to build publicly available portals. 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.

In this book, you will learn about the various components of 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. You will also learn what it means to set up an enterprise implementation project to successfully implement a Power Platform solution for a large organization.

While reading the book, you will accompany a fictitious global corporation, Contoso Inc., on their journey into the Power Platform cloud. You will see how they are adopting the products, preparing the implementation project, learning all the aspects of such a complex solution, and taking decisions in order to reap the largest possible rewards, all while following internal policies and rules.

After reading this book, you should have a clear understanding of the Power Platform technology and know how to use it to deliver a robust, scalable, and user-friendly solution to your customers.

Who this book is for

This book is primarily intended for solution and technical architects who would like to understand all aspects of a successful Power Platform project implementation. But, of course, every technical specialist on the project implementation side, as well as on the customer side, who wants to learn about Power Platform will benefit from reading this book.

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

What this book covers

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

Chapter 2, Microsoft 365 and Microsoft Azure Overview, provides 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 the context of a Power Platform solution, and how the solution can benefit when integrating them. You will also get an overview of how those cloud components are licensed.

Chapter 3, Understanding Microsoft Power Platform Architecture, dives deep into the world of the 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. You will also focus on the administration and monitoring possibilities, and finally, you will learn about various architectural best practices that are proven for Power Platform implementations in large organizations.

Chapter 4, Tools and Techniques, covers the tools and techniques used to configure and customize Power Platform solutions. You will also learn about the tools and techniques that need to be used when configuration and customization do not cover advanced customer requirements, and custom development is necessary. Finally, you will learn about some of the tools supporting the application lifecycle of Power Platform solutions.

Chapter 5, Application Lifecycle Management, explains how to adopt Application Lifecycle Management (ALM) principles to implement Power Platform solutions. You will learn about solution management as the main ALM approach for all parts of a solution except for Power BI. You will look at the potential application of ALM in Power BI. You will also learn how Microsoft Azure DevOps can help you make the ALM processes smooth, easy, and fully automated. At the end of the chapter, you will be presented with a collection of ALM-related best practices.

Chapter 6, Implementation Approach and Methodologies, covers a lot of the practical details about how to understand your customer's enterprise architecture. You will see what project implementation methodologies are available, which ones are used for Power Platform projects, and how to prepare an implementation project. Finally, you will see what a typical project setup in terms of roles and responsibilities can be, and what an enterprise project lifecycle, from the beginning until the solution is brought into production, can look like.

Chapter 7, Microsoft Power Platform Security, focuses on all aspects of Power Platform solutions security. You will learn all the details of authentication within Microsoft cloud solutions, and you will look at the details in relation to Power Platform. You will also see how authorization in the various Power Platform components can be implemented. At the end of the chapter, you will learn a number of security-related best practices.

Chapter 8, Microsoft Power Platform Extensibility, dives deep into the extensibility possibilities of the various Power Platform components. You will learn what can be achieved by configuration and customization and what types of requirements need to be developed with code. The main part of this chapter is dedicated to Common Data Service applications, but you will also gain an understanding of the extensibility capabilities of canvas apps, Power Automate, and Power BI. At the end of the chapter, various extensibility best practices will be presented.

Chapter 9, Microsoft Power Platform Integration, explains that every complex Power Platform solution is always heavily integrated with other Microsoft cloud services, but also with the customer's own IT ecosystem. That's why it is crucial to understand the integration possibilities of Power Platform. In this chapter, you will learn about all those Microsoft 365 and Microsoft Azure integration options, and how lot things can be achieved by using a simple configuration. You will also see how a custom frontend and backend integration can be implemented and explore the typical integration patterns and solution approaches. Additionally, we will explore several integration capabilities of Power Virtual Agent, AI Build, and Power BI.

Chapter 10, Microsoft Power Platform Data Migration, is dedicated to data migration. In most cases, the implementation of a Power Platform solution must be accompanied by a data migration effort to bring all the useful data from the customer's various legacy IT systems and solutions into Power Platform. In this chapter, you will learn what the usual data migration strategies are and what tools and solutions can be used for this purpose. At the end of the chapter, you will be presented with best practices to mitigate the different challenges of a complex 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. Also, a number of tools are required to perform configuration, customization, and custom development on Power Platform. These tools are described in detail in *Chapter 4, Tools and Techniques*.

The best environment to successfully use the concepts presented in this book is a Windows-based PC with a fast internet connection and any of the major internet browsers.

If you are using the digital version of this book, we advise you to type the code yourself or access the code via the GitHub repository (link available in the next section). Doing so will help you avoid any potential errors related to copying and pasting of code.

Download the example code files

You can download the example code files for this book from your account at www.packt.com. If you purchased this book elsewhere, you can visit www.packtpub.com/support and register to have the files emailed directly to you.

You can download the code files by following these steps:

1. Log in or register at www.packt.com.
2. Select the **Support** tab.
3. Click on **Code Downloads**.
4. Enter the name of the book in the **Search** box and follow the onscreen instructions.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR/7-Zip for Windows
- Zipeg/iZip/UnRarX for Mac
- 7-Zip/PeaZip for Linux

The code bundle for the book is also hosted on GitHub at <https://github.com/PacktPublishing/Microsoft-Power-Platform-Enterprise-Architecture>. In case there's an update to the code, it will be updated on the existing GitHub repository.

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://static.packt-cdn.com/downloads/9781800204577_ColorImages.pdf.

Conventions used

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

Code in text: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "The workspaces module, which is used for workspace management (module name: `MicrosoftPowerBIMgmt.Workspaces`)."

A block of code is set as follows:

```
using System;
using Microsoft.Crm.Sdk.Messages;
using Microsoft.Xrm.Tooling.Connector;

static void Main(string[] args)
{
```

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

```
AuthenticationContext authContext = new
AuthenticationContext("https://login.microsoftonline.com/
common" + tenantid, false);
ClientCredential credential = new ClientCredential(clientId,
clientsecret);
```

Bold: Indicates a new term, an important word, or words that you see on screen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "The second option, **Stage for Upgrade**, provides an additional possibility to temporarily keep both the original and the updated solution in the environment."

Tips or important notes

Appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at customercare@packtpub.com.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit www.packtpub.com/support/errata, selecting your book, clicking on the Errata Submission Form link, and entering the details.

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@packt.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 authors.packtpub.com.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit packt.com.

Section 1: 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 clouds – Microsoft 365 and Microsoft Azure, – since a typical enterprise Power Platform solution is often successfully combined with the strengths and capabilities of the other two Microsoft clouds.

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

Microsoft Power Platform is a quickly growing collection of technologies, frameworks, solutions, and products. In this chapter, you will learn 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 full understanding of all components of the Power Platform and will enable you to draw a high-level overview of all the 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
- Practical example

Introducing Contoso Inc.

Contoso Inc. is a fictitious large global manufacturing and retail company with headquarters in Seattle, Washington, and a number of regional subsidiaries on all continents. Contoso Inc. works in the areas 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 themselves 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 mind 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 the following 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):

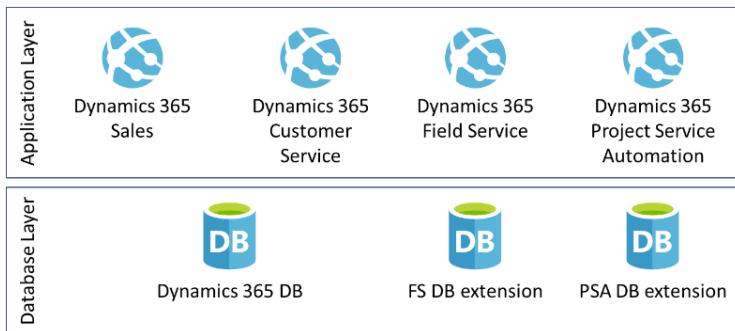


Figure 1.1 - Microsoft Dynamics 365 before the Power Platform

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

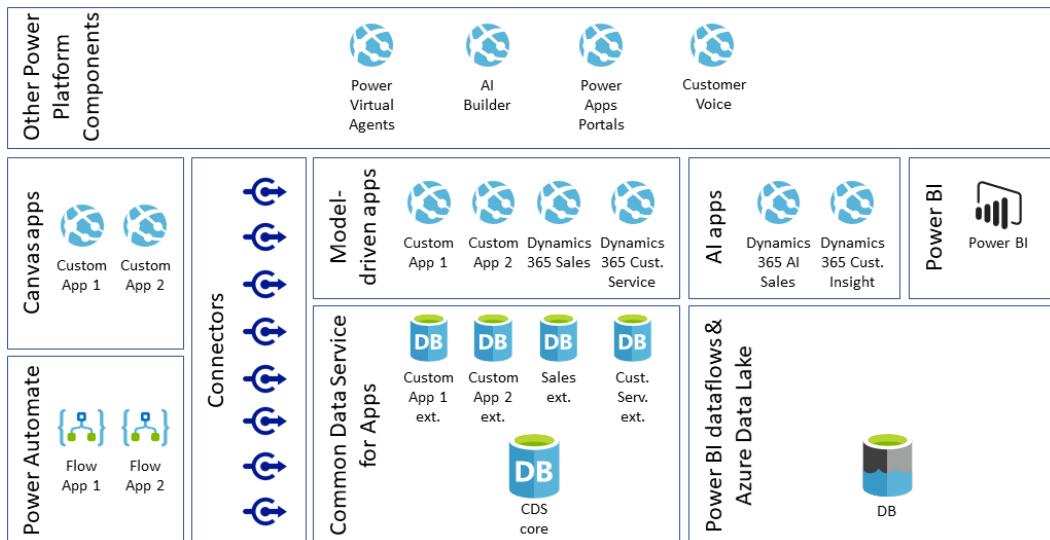


Figure 1.2 - Microsoft Power Platform

Microsoft Power Platform consists of the following components:

- **Common Data Service**
- **Model-driven apps**
- **Canvas apps**
- **Power Automate**
- **Power Virtual Agents**
- **Power BI**
- **On-Premises Data Gateway**
- **AI Builder**
- **Power Apps portals**
- **Dynamics 365 Customer Voice**

This new Microsoft Power Platform philosophy suggests that a potential user of a Microsoft-based business solution will need more evaluations and decision making to find the best solution for their business requirements. Some of the examples 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 for mobile, or do I need to develop my own apps using the *canvas apps* technology?
- Do I need to integrate 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 more than 350 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 Common Data Service is and what the new role of 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 looking at cross-platform automations and what the licensing options are for the various Power Platform components.

Introducing the Common Data Model and Common Data Service

Power Platform introduced the **Common Data Model (CDM)** and **Common Data Service (CDS)** and it is important to understand what these two concepts are and what the difference between them is.

Introducing 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 relatively small set of *core* entities (equivalent to tables in relational databases), which are not directly related to any particular workload, together with a lot of additional entities 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, and media.

Introducing Common Data Service

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

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

The CDS is the foundation for building model-driven apps. The approach is to first configure the whole CDS data model, create all elements, and then configure a model-driven app using the necessary subset of the CDS elements. Depending on the purchased licenses, the CDS is either extended 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 the CDS.
- There can be only one CDS 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 the CDS.
- Model-driven apps run primarily on a PC in a browser but can also be used on mobile devices within platform-specific mobile apps.

Important note

You will learn details of the Power Platform environment in *Chapter 3, Understanding 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
- Model-driven app navigation called the **Site Map**

All other components in a model-driven app are stored within the CDS 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 Apps environment with the CDS.
2. Create the data model (entities with fields 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 do not need the CDS; they can be created in a Power Apps environment without the CDS.
- Canvas apps are designed very much like apps for mobile devices with the user interface in focus.
- The business logic in canvas apps is implemented using *Excel-like expressions*.
- Canvas apps can be connected to various data sources using the concept of *connectors*. Right now, there are more than 350 publicly available connectors and there is a possibility to 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 a specific canvas apps player. They can run on a PC in a browser or can be embedded in websites, SharePoint sites, Power BI, Teams, or model-driven apps.

For the 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 the CDS.
2. Create a canvas app.
3. Optionally connect to data sources using *connectors*.
4. Create the user interface (for example, screens, galleries, forms, and controls).
5. Create the business logic using *expressions*.
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 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 on mobile devices within a specific Power Automate player app.
- **Scheduled flows:** These are triggered in the background by a timer or scheduler trigger.
- **UI flows:** These are used for recording and automating manual steps on various legacy software.

For the 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 the CDS.
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 bots using a *graphical interface* and a *no-code approach* and so open the world of bots to everyday business users without specific programming skills. The following are the capabilities of Power Virtual Agents:

- Bots are developed in a graphical designer with no code requirements.
- The designer defines conversational topics, business logic, and actions for the conversation. The actions can be implemented using Power Automate.
- The bot 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 bot can be divided into the following steps:

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

Introducing Power BI

Power BI is a collection of a data platform, cloud services, applications for PC and for mobile devices, and connectors to work 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:

- 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 the CDS 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 Apps 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 your own on-premise data sources within various cloud services such as Power Apps, Power Automate, and Power BI but also for some Microsoft Azure services. On-Premises Data Gateway needs to be installed on a local infrastructure and configured to expose the required on-premise 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* and so to open 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 with no code requirements.
- The user selects one of the ready-made AI models the platform offers, provides data for training the model, trains, and publishes the solution.
- The AI solution can be used from Power Automate flows as well as from canvas apps to infuse AI-processed content into an automation or mobile application.

Introducing Power Apps portals

Power Apps portals historically evolved from *Microsoft Dynamics 365 portals*. The purpose of **Power Apps portals** is to provide external-facing websites connected with the CDS data for users outside of their own organization. Power Apps portals are the only Microsoft Power Platform technology that is really open to public and even anonymous access. Power Apps 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 Apps portals expose selected data from the CDS to anonymous or registered and authenticated users.
- Power Apps portals offer a wide variety of authentication possibilities for external visitors.

Introducing Dynamics 365 Customer Voice

Dynamics 365 Customer Voice is a solution for creating and running surveys and is based on the Microsoft Office 365 Forms technology for the UI part and the CDS for the content and data part. Dynamics 365 Customer Voice has the following capabilities:

- Design surveys in a graphical designer using different question types and branching logic.
- Distribute surveys to participants in a variety of ways including email, Power Automate, by embedding in a web page, providing a survey link, or with a QR code.
- Combine responses with CDS business data and analyze them using included Power BI analytics.

Introducing Microsoft Dynamics 365 CRM applications

The **Microsoft Dynamics 365 CRM technology** is historically the foundation for the CDS, but in the current concept, the CDS is the foundation and the Microsoft Dynamics 365 CRM applications are called CDS **first-party** apps. Each of the applications, when provisioned, extends the CDS 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-premise 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 (implemented in a separate app Dynamics 365 Customer Voice)
- Reporting and analytics

Microsoft Dynamics 365 Marketing

The **Microsoft Dynamics 365 Marketing** app has the following capabilities:

- Customer journey
- Marketing content
- Segments

- Marketing emails
- Marketing forms
- Marketing pages
- Lead management
- Internet marketing
- Social media marketing
- Event management
- Surveys (implemented in a separate app Dynamics 365 Customer Voice)
- 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
- Omnichannel add-on
- Reporting and analytics

Microsoft Dynamics 365 Field Service

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

- Work order management
- Resource management
- Field service resource scheduling
- Agreements
- Inventory management

- Asset management
- Field service mobile
- Connected field service add-on
- Reporting and analytics

The Dynamics 365 Connected Field Service *add-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, except for **Dynamics 365 Human Resources**, are not based on the CDS platform and are not considered model-driven applications. Some of the applications are available also as an on-premise 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 most modern technologies such as artificial intelligence and mixed reality. The result is a large and always growing number of Dynamics 365 add-ons 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
- Customer segmentation
- 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, of which some are included in the Sales app for free, while others are based on additional licenses or even included in a separate application:

Free with Dynamics 365 Sales	Advanced features for Dynamics 365 Sales	Dynamics 365 Sales Insights App
<ul style="list-style-type: none"> • (Relationship) Assistant • Auto Capture • Email Engagement 	<ul style="list-style-type: none"> • Advanced Assistant • Relationship Analytics • Predictive Lead Scoring • Predictive Opportunity Scoring • Notes Analysis • Who Knows Whom • Talking Points • Assistant App for Teams 	<ul style="list-style-type: none"> • Sales Analytics Overview • Team Analytics Overview • Seller Analytics Overview • Natural language Q&A

Figure 1.3 - Dynamics 365 Sales Insights capabilities

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

Microsoft Dynamics 365 Customer Service Insights

Microsoft Dynamics 365 Customer Service Insights is a Power-BI-based solution, providing analytical and AI-enriched insights into key customer service data, KPIs, and metrics:

- Case overview and breakdown
- Topic-based case grouping
- Current and emerging trends in customer service
- New case analytics
- Customer satisfaction analytics
- Case resolution analytics

Microsoft Dynamics 365 Product Insights

Microsoft Dynamics 365 Product Insights is an *IoT*-and *AI-based* solution for analyzing signals coming from sold products, providing AI-driven analytics and Power-BI-driven reporting capabilities. The product signals can be everything from data collected online from various information systems or data imported from files, to data collected from IoT connected sensors. The solution provides developer resources for developing integration between signal-producing equipment and Microsoft Dynamics 365 Product Insights. The final collected and analyzed data is presented using the analytical and reporting capabilities of Microsoft Power BI.

Microsoft Dynamics 365 Connected Store

Microsoft Dynamics 365 Connected Store 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 needing attention. The solution analyzes 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 in 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 uses *HoloLens*, 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 iOS mobile devices only. The solution is directly integrated with Microsoft Dynamics 365 Sales and Microsoft SharePoint.

Microsoft Dynamics 365 Import Tool

Microsoft Dynamics 365 Import Tool is an on-premise tool supporting the import of 3D assets into mixed reality apps, Microsoft Dynamics 365 Guides, and Microsoft Dynamics 365 Product Visualize.

Microsoft Dynamics 365 Unified Service Desk

Microsoft Dynamics 365 Unified Service Desk is a framework for building call-center applications, integrating CDS applications with existing legacy and third-party applications of various types (web, desktop, Java, and mainframe), 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:

Licensing	Description	Price
Trials	For almost all Microsoft Power Platform components there is a possibility to provision a free trial usually limited to 30 days	Free
Power Apps Community Plan	Specific free of charge permanent license dedicated to learning and individual development, only 1 single user is supported	Free
Power Apps / Power Automate for Office 365	These licenses are included in most Office 365 subscriptions and allow the creation of Power Apps and Power Automate flows based on Office 365 services	Included in Office 365
Power Apps / Power Automate for Dynamics 365	These licenses are included in most Dynamics 365 subscriptions and allow the creation of Power Apps and Power Automate flows based on the Dynamics 365 licensed products	Included in Dynamics 365
Power Apps standalone plans	These licenses contain Common Data Service and allow the creation of Power Apps beyond Office 365	\$10 - \$40 / user / month
Power Automate standalone plans	These licenses contain Common Data Service and allow the creation of Power Automate flows beyond Office 365	\$15 / user / month or \$500 / 5 flows
Power Virtual Agents	Power Virtual Agents are licensed per tenant and the price is based on bot sessions	\$1,000 / 2,000 sessions
Power Apps Portals	Power Apps portals are licensed based on a consumption model per login and per page view	
AI Builder	AI Builder is licensed based on units of 1 million service credits. Service credits are consumed when AI models are trained or used in Power Platform solutions	\$500 / unit

Power BI	<p>Power BI has three licensing options:</p> <ul style="list-style-type: none"> • Power BI Free is free of charge • Power BI Pro is licensed based on user / month • Power BI Premium is licensed based on dedicated resource 	<p>\$10 / user / month</p> <p>\$5,000 / resource / month</p>
Forms Pro for Office 365	Forms Pro is licensed as an upgrade for Office 365 users	\$100 / 2,000 survey responses
Forms Pro for Dynamics 365	Forms Pro is included in selected Dynamics 365 licenses	Free / 2,000 survey responses
Dynamics 365	Most Dynamics 365 products (Sales, Sales Insights, Customer Service, Customer Service Insights, Field Service, Remote Assist, Guides, Finance, Supply Chain Management, Project Service Automation, Retail, Business Central) are licensed per user / per month	\$20 - \$180 / user / month
Dynamics 365	Some Dynamics 365 products (Marketing, Customer Insights, Fraud Protection) are licensed per tenant / per month	\$750 – \$10,000 / tenant / month
Capacity	Every Microsoft Power Platform product includes a certain amount of free database storage capacity. When exceeded, additional storage capacity can be purchased.	

Figure 1.4 - Microsoft Power Platform licensing overview

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

For example, when 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 my requirements are covered as standard?

Let's answer these types of questions with a short and simplified 3-year cost calculation.

Simple relationship management

In the following table, we provide a high-level simplified cost calculation, where for a Dynamics 365 Sales Professional application (\$65/user/month) we calculate with 80 hours of implementation of the out-of-the-box product, for the Power Apps Per-App Plan (\$10/user/app/month) with 320 hours of implementation including light customization. The cost of labor is estimated at \$150/hour:

	Dynamics 365 Sales license	Power Apps single-app license
License costs 3 years for 100 users	\$234,000	\$36,000
Implementation	\$12,000	\$48,000
Total cost	\$246,000	\$84,000
Monthly cost	\$6,830	\$2,330

Figure 1.5 - Sales Management Cost comparison example

As we can see in this case, purchasing a Power Apps license is significantly beneficial over Dynamics 365.

Complex marketing management

In the following table, we provide a high-level simplified cost calculation where, for Dynamics 365 Marketing, we calculate 160 hours of implementation of the out-of-the-box product, and for Power Apps Per-App Plan (\$10/user/app/month), 5,000 hours of implementation including complex customization and custom development. The cost of labor is estimated at \$150/hour:

	Dynamics 365 Marketing license	Power Apps single-app license
License costs 3 years for 100 users	\$54,000 (licensed per instance not user)	\$36,000 (licensed per user)
Implementation	\$24,000	\$750,000
Total cost	\$78,000	\$786,000
Monthly cost	\$2,170	\$21,830

Figure 1.6 - Marketing Management Cost comparison example

As we can see in this case, a custom development approach would cost 10 times more than deciding on the right Dynamics 365 product providing the required capabilities.

Note

For a detailed information about Power Apps and Microsoft Power Automate licensing, please refer to the following link: <https://docs.microsoft.com/en-us/power-platform/admin/pricing-billing-skus>

Further details on the Power Platform components and Products are available at <https://powerplatform.microsoft.com/en-us/>

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 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 table:

Component	Usage Scenario
Dynamics 365 Sales, Sales Insights, Product Visualize	Contoso Inc. sales departments managing B2B sales processes and insights
Dynamics 365 Marketing, Power Apps Portals	Contoso Inc. marketing department running product campaigns and product presentation and conferences
Dynamics 365 Customer Service, Customer Service Insights, Unified Service Desk, Power Apps Portals, Power Virtual Agents	Contoso Inc. customer service departments managing service inquiries and providing remote help and support
Dynamics 365 Field Service, Connected Field Service, Remote Assist	Contoso Inc. field service department managing maintenance and repair onsite visits and proactive maintenance for customer's installed machinery equipped with IoT connected sensors
Dynamics 365 Project Service Automation	Contoso Inc. project implementation department managing onsite installation projects of sold machinery and factories
Dynamics 365 Finance	Contoso Inc. finance department managing finance accounting of every legal entity in the group.
Dynamics 365 Supply Chain management	Contoso Inc. production, procurement, logistics, and finance departments managing the respective business workloads.
Dynamics 365 Commerce, Connected Store, Fraud Protection	Contoso Inc. retail department managing retail stores and e-commerce sales.
Dynamics 365 Human Resources, Guides	Contoso Inc. human resources department managing all aspects of hiring, people management, benefits, training, readiness, and so on.

Dynamics 365 Customer Insights & Market Insights	Contoso Inc. sales organization managing 360° customer and competitor view including all external and internal information sources
Dynamic 365 Product Insights	Contoso Inc. managing analytical view on sold products for early recognizing trends and developments leading to continuous product improvement.
Canvas Apps	Contoso Inc. power users and citizen developers developing small single-purpose mobile apps
Power Automate	Contoso Inc. power users and citizen developers developing automations across the solution components
Power BI Premium	Contoso Inc. power users and citizen developers developing analytical and reporting solutions. To achieve advanced capabilities, Contoso Inc. will purchase Power BI Premium with appropriate premium capacity
Dynamics 365 Customer Voice	Contoso Inc. sales, customer service, and field service departments running surveys to improve customer satisfaction

Figure 1.7 - Overview of the Contoso's Power Platform commitment

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 of 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 clouds, 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 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 for a Microsoft Power Platform solution to better understand the added value of the combined power of the three Microsoft clouds.

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 afforded by the other two Microsoft cloud offerings in order to enhance the capabilities of the Power Platform, to integrate the Power Platform with other useful cloud services, and to 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 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 for the Power Platform.

Introducing Microsoft 365

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

Microsoft 365 consists of three main groups of products:

- **Microsoft Windows 10**
- **Microsoft Office 365**
- **Microsoft Enterprise Mobility + Security**

You'll already be familiar with the purpose of the Windows 10 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 the cloud messaging solution based on the established on-premise product. Microsoft Exchange is one of the most integrated Microsoft Server products in Power Platform solutions, since it extends the solution with the email channel, and supports the calendar, appointments, tasks, and contact synchronization. The integration with Exchange is also required for Power Platform integration with Office 365 groups.

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**, **Microsoft Teams**, and **Office 365** groups.

Microsoft OneDrive for Business

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

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 and video, and attached files. Power Platform solutions integrate with OneNote to provide a context-based, note-taking capability.

Microsoft Teams

Microsoft Teams is the new 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 Yammer

Microsoft Yammer is an enterprise social network solution providing similar capabilities to Teams. Yammer is used primarily for chat-based communication, but also supports document sharing. Power Platform solutions integrate with Yammer primarily for providing the context-based chat capability.

Microsoft Office 365 Groups

Microsoft Office 365 Groups is another collaboration solution offering email communication, group calendars, file sharing, and integration with other apps. Power Platform solutions integrate with Office 365 Groups to provide a collaborative workspace for users not having a Power Platform product license.

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 is a group of cloud and on-premise products for security and mobility management. The group consist of various products, not all of which are relevant for 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-premise solution dedicated to the 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 Dynamics 365 client for Outlook, 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.

Using Microsoft Exchange

Contoso Inc. plans to establish server-side integration between the Dynamics 365 applications and Exchange so as to provide an integrated email and calendar experience.

Using Microsoft SharePoint

Contoso Inc. will establish integration between the Dynamics 365 applications and SharePoint for integrated document management, specifically in the sales department.

Using Microsoft OneDrive for Business

Contoso Inc. will establish integration between the Dynamics 365 applications and OneDrive for Business so as to enable a private document store for confidential documents where sharing is not desired.

Using 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 context with Dynamics 365 records.

Using 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.

Using Microsoft Intune

Contoso Inc. will use Intune to deploy and manage the Power Platform mobile apps on employees' mobile devices.

Using 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 solution.

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 of **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.

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.

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.

As well as AAD, there is also a component called **Azure Active Directory B2C (Business to Consumer)**, used to manage external identities such as customers, consumers, and citizens. AAD B2C has a special relevance for Power Apps portals, where it can be used as one of the options for the 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. Power Platform contains a built-in integration with Azure Service Bus to support integration solutions. The solution can be used to perform near real-time outbound integration from a Power Platform solution to the customer's on-premise data center or to another cloud solution.

Introducing Azure Event Hub

Azure Event Hub 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 Hub 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 more than 350 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-premise 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 the 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 CDS database into its own Azure SQL instance to get full access to Power Platform data on the database level. This capability can be achieved by using the free Microsoft *Data Export Service* AppSource module.
- Database consolidation to consolidate data from CDS 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 entities 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 several 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 CDS database to a much less expensive storage type, for example, using the free Microsoft *Attachment Management* AppSource module.

Introducing Azure Data Lake Storage Gen2

Azure Data Lake Storage Gen2 is a cloud platform for building enterprise data lakes, scaling to multiple petabytes and high throughput. Power Platform has a built-in integration with Azure Data Lake Storage Gen2 for the continuous replication of CDS data. The service is fully configurable and enables the use of CDS data in the data lake for analytical and reporting purposes with Power BI and machine learning.

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 both 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.

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*.

Introducing Azure Monitor

Azure Monitor is a collection of telemetry and monitoring tools to analyze the cloud as well as on-premise 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.

Using authentication and single sign-on

Contoso Inc. has recognized that using cloud identity for the current Microsoft 365 subscription being deployed is just a temporary solution. They decided, for the purpose of full governance and true single sign-on capability, that they would deploy the Azure Active Directory Federation using *Azure Active Directory Connect*. To authenticate their external portal users, they will implement *Azure Active Directory B2C*.

Power Platform integration

Contoso Inc. has analyzed the existing IT landscape and identified multiple existing special purpose on-premise, 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 Hub
- 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 lifecycle 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 the 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.

Understanding 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, 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 in various ways, as well as Office 365 and EM+S subscriptions of different types.

Note

Further details about Microsoft 365 licensing can be found in the following website: <https://www.microsoft.com/en-us/licensing/product-licensing/microsoft-365-enterprise>

Learning about 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 use certain Azure resources for a period of 1 year or 3 years.

Note

For more details Azure licensing, please refer to the following link:
<https://azure.microsoft.com/en-us/pricing/>

Summary

In this chapter, you have learned the basics of the two other Microsoft clouds: Microsoft 365 and Microsoft Azure, which are used heavily in Power Platform solutions. Armed with this knowledge, you should be able to assess the feasibility of various Microsoft 365 and Microsoft Azure components when you start architecting your Power Platform solution for security, artificial intelligence, IoT, integration into your existing IT landscape, data migration, and many more applications besides.

In the next chapter, we will dive deeper into the Power Platform architecture, specifically, the Power Platform environments and structure, clients, administration, and monitoring.

Section 2: 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 Microsoft Power Platform Architecture

Chapter 4, Tools and Techniques

Chapter 5, Application Lifecycle Management

3

Understanding Microsoft's 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 design 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 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 three Microsoft cloud offerings; that is, **Power Platform**, **Microsoft 365**, and **Microsoft Azure**, Contoso Inc. has made a series of high-level decisions with regards to which products will be part of their future solution. Now is the time to start architecting their planned Power Platform solution, and 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 offerings 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, 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 of all three clouds 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 more than 50 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.

However, for **Power Platform**, there's currently just three separately selectable regions in Europe; namely, Europe, United Kingdom, and France. This indicates that Power Platform cloud services are not available in all existing Microsoft data centers.

For Power Platform cloud services, the customer cannot select the region in such 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 (the number of available regions can grow in the future):

- Asia
- Australia
- Canada
- Europe
- France
- India
- Japan
- South America
- 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 available in the other regions. Selecting this region is useful for testing the latest product features and giving feedback to Microsoft. Currently, no **Common Data Service (CDS)** database can be created in this region, so no CDS and model-driven app preview features are available when using this channel.

Understanding the customer cloud structure

When a customer first purchases a 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 environment, there can exist several subscriptions of cloud services from all three Microsoft clouds. 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:

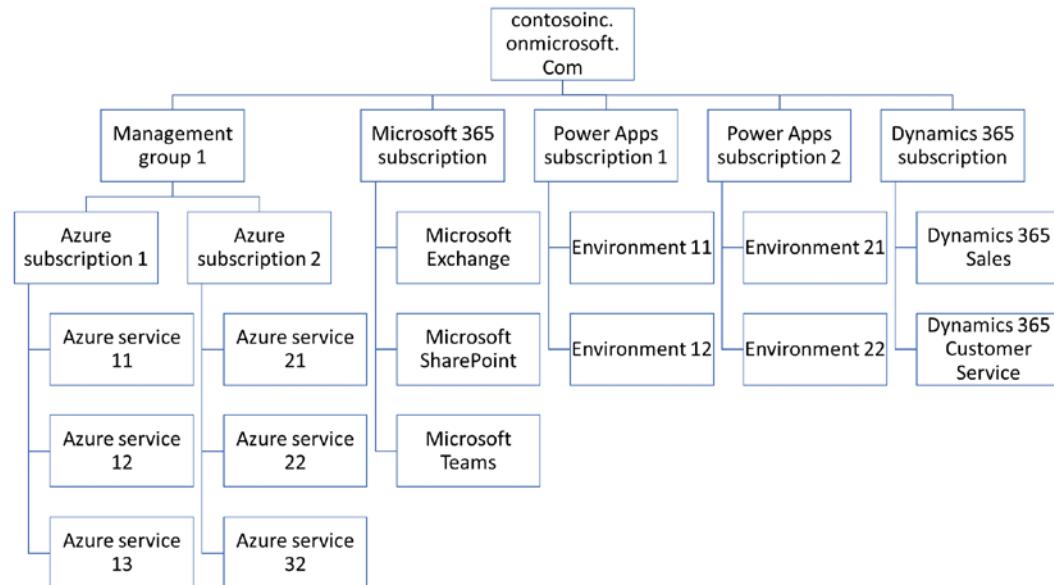


Figure 3.1 - Contoso Inc. structure of the Microsoft cloud services

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, the central tenant services that are essential for running any Power Platform application are 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 management 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 registered 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 CDS applications
- Managing the integrated Office 365 groups

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

App registrations

App registrations 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 CDS 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 services. There are database servers, reporting servers, web servers, app servers, integration servers, and much 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. 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. For example, customers aren't allowed to run stress tests against the Power Platform API. 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 (other than **Power BI**) always run within Power Platform environments. An environment is a logical and physical 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 six 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.
- **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 community plan license. This environment can have only the owner as a single user.
- **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.
- **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, every environment has some basic parameters that must be specified upon creation:

- **The display name** of the environment
- **Environment type**, which can be either a **Sandbox**, **Production**, or **Trial** environment
- **Region**, which is selected from a list of Power Platform-enabled regions
- **The purpose** of the environment as a free text description

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

If a CDS database should be created for the new environment, then the following additional parameters need to be specified:

- **Language**: This is the base user interface language of the CDS database and is used in all model-driven applications created in this CDS.
- **Currency**: This is the base currency of the CDS database used by currency data type fields.
- You need to either enable the deployment of some Dynamics 365 applications or a few sample model-driven applications with sample data.

If a CDS 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 CDS 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.

A Power Platform environment consists of the following main components:

- **CDS** and its components
- **Power Platform connectors**

- **Data Loss Prevention (DLP) policies**
- **On-premises data gateway**

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

Common Data Service

Common Data Service is used to store the metadata, business data, and application artifacts of model-driven apps. The CDS storage is subdivided into the following three storage types:

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

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 number of user licenses a customer has purchased. Currently, the basic storage capacity every new customer gets with their first subscription is as follows:

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

The following environments are excluded from the storage capacity limits:

- Trial
- Preview
- Support
- Developer

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

Tip

Microsoft offers the possibility to purchase additional storage capacity add-ons to increase the available storage within the tenant.

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

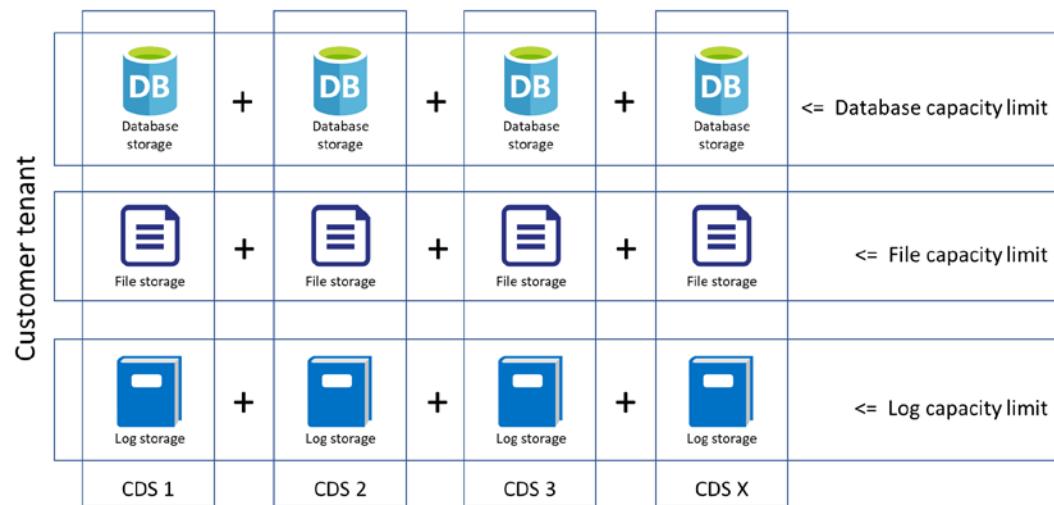


Figure 3.2 - Storage capacity limits

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.

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 CDS solution within a 24-hour period. Requests that count toward these limits can come from the following:

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

The values of the request limits depend on the license type the user communicating with the platform has. These vary between 1,000 and 20,000 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/powerapps/developer/common-data-service/api-limits>.

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

Power Platform connectors

Power Platform connectors are basically wrappers around certain APIs provided by various Microsoft and non-Microsoft services. These connectors allow us to connect to services from **canvas apps**, **Power Automate flows**, and **Azure Logic Apps**. 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 Apps 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.

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

- **Tables** are available for certain connectors in canvas apps and make it possible to connect the data source behind the connector with canvas apps controls (gallery, data card, and so on). The CDS connector is an example of a connector that provides the *Tables* property.
- **Triggers** are available for certain connectors for Power Automate flows to start the flow of execution. For CDS connector, for example, there are triggers such as the following: **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.
- **Actions** are available for certain connectors for both canvas apps and Power Automate flows and are basically executions of changes in data or other manipulations, performed by the connector from within a canvas app or Power Automate flow. For CDS connector, for example, there are actions such as **Create record**, **Update record**, **Delete record**, **Get record**, and many others available.

DLP policies

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 Power Platform environments in the tenant.
- **Environment level:** This is valid only in the respective selected environments, or in all tenant environments except the selected one. These policies cannot override the tenant-level policies.

These policies define rules for what connectors can be used together in a solution to effectively block their possible dangerous exposure. The DLP policy can place these 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 the DLP configuration will require putting the selected sensible connectors into this group.
- **Non-business data:** This is the default group that all the connectors are initially part of.
- **Blocked:** Connectors in this group are entirely blocked within the specified scope and cannot be used in canvas apps and Power Automate flows.

By default, no DPL policies are created in a new Power Platform ecosystem. It is important to understand that a combination of multiple DLP policies always leads to the most restrictive result.

On-premises data gateway

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

- Power Apps
- Power Automate
- Power BI
- Azure services (Azure Logic Apps, 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**

This is illustrated in the following diagram:

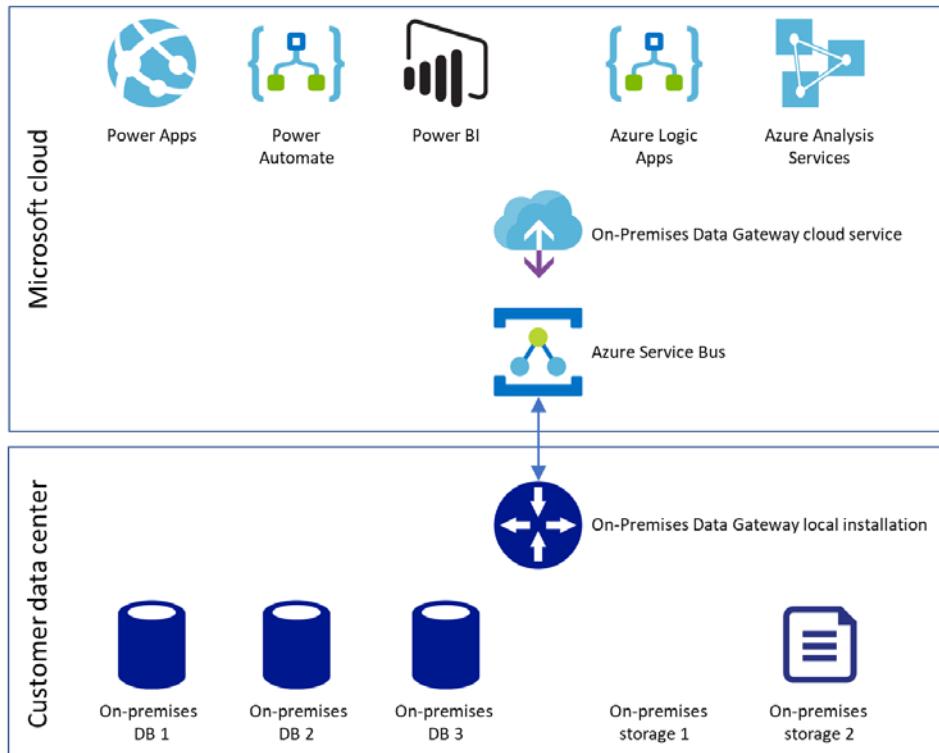


Figure 3.3 - On-premises data gateway architecture

The respective components of the gateway are responsible for encrypting and decrypting the on-premises data source's credentials, 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 database to cloud connection without the need to open any inbound communication, thus exposing the internal network to potential attacks guided by malicious players.

Now, let's learn about Power BI's structure.

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:

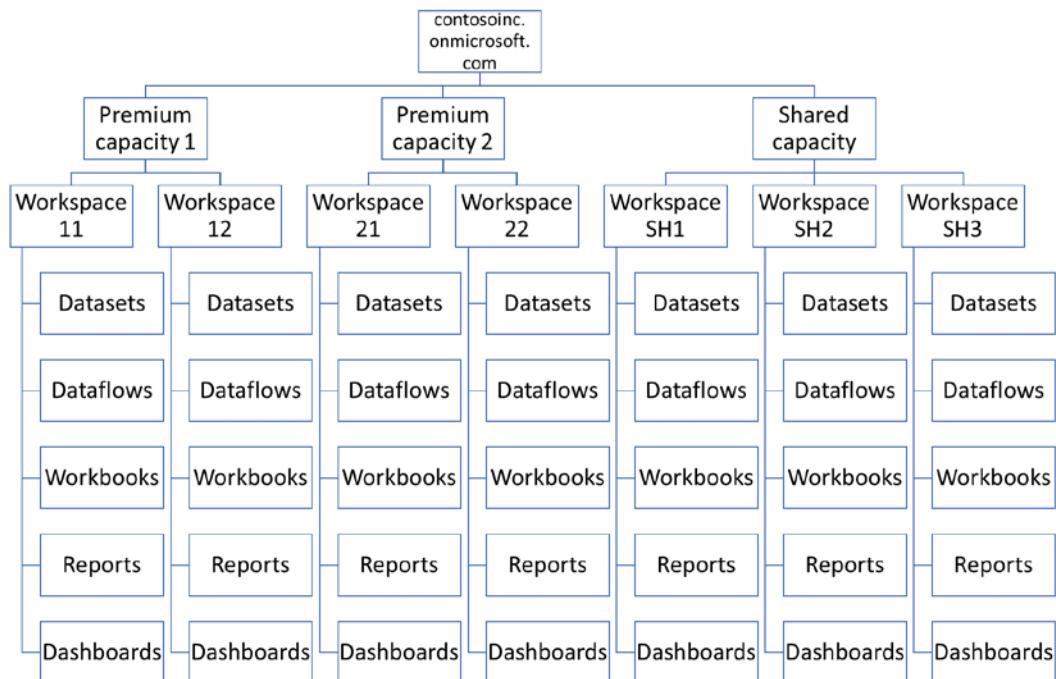


Figure 3.4 - Contoso Inc. example structure of the different Power BI services

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).
- **Datasets** are data collections used directly as data sources for Power BI reports.
- **Dataflows** are data sources prepared for pushing into datasets.
- **Workbooks** are specific datasets based on Excel.

- **Reports** are the main visualization objects created in Power BI using one single dataset.
- **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, along with 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 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 OSes. In the following sections, you will learn about the 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 OS combinations are supported:

- **Internet Explorer 11**
- **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)

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.

Important note

The legacy desktop Microsoft Dynamics 365 client for Outlook is still available for historical reasons. This client application is deprecated and should not be used for new Dynamics 365 implementations. This client has, however, certain unique features that aren't available in the modern client app yet – specifically, full offline capabilities, the ability to integrate with multiple Dynamics 365 instances, and client-side synchronization.

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

Unified Service Desk

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

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

Common Data Service solution

The CDS is used to host the USD entities 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 the contact center/call center agents for their daily business. The agent logs into the agent desktop, and then Agent Desktop connects to the CDS environment from which the configuration for the business processes is then retrieved. If the business functionality changes, this change can be done centrally within the configuration of the underlying CDS solution.

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

- Integration and automation of applications
- Session management
- Hosting various application types (hosted controls, web application, **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*

Part of Power Platform are two technologies; namely, **Robotic Process Automation with UI Flows** and **Omnichannel for Dynamics 365 Customer Service**. These partially cover the capabilities of USD with less overhead and no need for local deployments.

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

Robotic Process Automation with UI Flows

UI Flows are the latest members of the Power Automate technology. UI 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.

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:

- 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**

The benefit of this app is that you can extend Dynamics 365 Customer Service with many important communication channels, without the need for custom development or deploying any local software such as USD.

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 apps provide a selection of all model-driven apps a user has access to and allow us to start any of those apps. The user interface's design and look and feel is identical to the design for the browser client, just optimized for the small form factor.

Important note

The aforementioned mobile apps provide limited offline capability. This capability requires certain mobile offline configuration.

- **Power Apps mobile player:** This is a mobile client for canvas apps. The app provides a selection of all canvas apps a user has access to and allows us to start any of those apps.
- A unified mobile client app to run both model-driven apps, as well as canvas apps in one environment, is currently in preview and will replace the Dynamics 365 app and the Power Apps app.
- The **Field Service mobile app** is a Microsoft-branded third-party mobile application, used for **Dynamics 365 Field Service**. This is due to historical reasons, since Dynamics 365 Field Service was acquired by Microsoft from a partner company called **FieldOne**, and their module was integrated with this third-party mobile application. Field Service Mobile provides certain features not yet available in the native Dynamics 365 app for phones and tablets.

Important note

Microsoft provides a new CDS based Field Service (Dynamics 365) app as a preview offering. This new app will replace the third-party mobile app until summer 2021.

- **Power Automate mobile app:** 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 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 and reports 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 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 achieving the necessary level of governance.

It is increasingly important, with the growing complexity of your own ecosystem, to ensure governance, so that you can 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 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 three 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 365 admin center

Microsoft 365 admin center is used for administration purposes regarding the Microsoft 365 central 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 **SaaS** cloud services.

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:

- **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.

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

The **Power Platform admin center** is the main administration center and is an entry point with links to other underlying administration centers. 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 encompass 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 about 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**
- **Links to the other administration centers**

The 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 the administration and monitoring purposes.

Capacity analytics

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

Common Data Service analytics

Common Data Service analytics provides detailed insights into the following areas of the CDS 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)
- **Entity usage** analytics (most used entities and custom entities)
- **System jobs, plugin, API usage**, and mailbox usage analytics

Power Automate analytics

Power Automate analytics provides detailed insights into the following areas:

- **Flows runs** analytics (daily, weekly, monthly)
- **Flows 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.

Learning about the Power BI admin center

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

This admin center 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 the Windows PowerShell module is planned for future deprecation once its capabilities are fully available in Azure Active Directory PowerShell for Graph.

One of the interesting options would be to create a set of PowerShell scripts such as the previous one and perform user, groups, and licenses management in an automated way.

Power Apps administration

There are multiple PowerShell modules for administering Power Apps, as well as Dynamics 365. The following are the three modules for Power Apps administration:

- The module for **Power Apps administrators** (module name: `Microsoft.PowerApps.Administration.PowerShell`) provides the following capabilities:

CDS 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 last module (module name: `Microsoft.PowerApps.Checker.PowerShell`) is used to manage the **Power App checker service**.

There is another group of three modules for administering Dynamics 365 instances, as follows:

- The module for administering Dynamics 365 instances and **Bring Your Own Key (BYOK)** encryption keys (module name: `Microsoft.Xrm.OnlineManagementAPI`)
- The module for connecting to Dynamics 365 instances and retrieving instance 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 CDS database using the `Microsoft.PowerApps.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 region of the United States with an environment type of **Production**. It is specified that a CDS database will be created for this environment. For environments created with the CDS database, the primary currency and primary language are mandatory parameters.

This example illustrates one possible approach to administering Power Platform; that is, 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, 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 generating 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's 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, the 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.

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

The **Microsoft Graph API** is used for Microsoft 365 administration with code. The **Microsoft Graph API** has a rich set of programmability models for managing Office 365, Windows 10, as well as Enterprise Mobility and Security. The 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 a 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 Apps administration

Administering CDS with code can be performed using the REST-based **Online Management API** for CDS. This service is provided using a region-dependent URL: <https://admin.services.crmX.dynamics.com>.

The X in the URL is the Power Platform region code.

This API supports most of the operations that are available in the admin centers:

- General environment/instance management (create, configure, update, delete, backup, restore, get instance information)
- Management of encryption keys
- Notification management
- Management of application identities
- Testing management

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 shown in the following diagram:

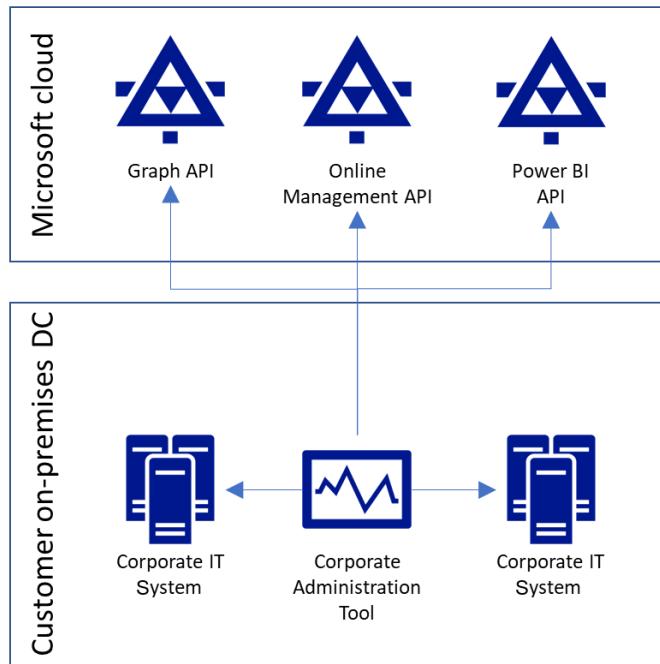


Figure 3.5 - Centralized API-based management solution

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 grant 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 administration of Power Platform cloud environments and the Power BI instance. This can be done by using the Online Management 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 with **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 approval purposes.
- **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 users and groups 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, managing permissions, and Power Apps.
- **Power Apps for App Makers connector**, which can be used for retrieving information about Power Apps, managing permissions, Power 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 and refreshing a dataset.

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:

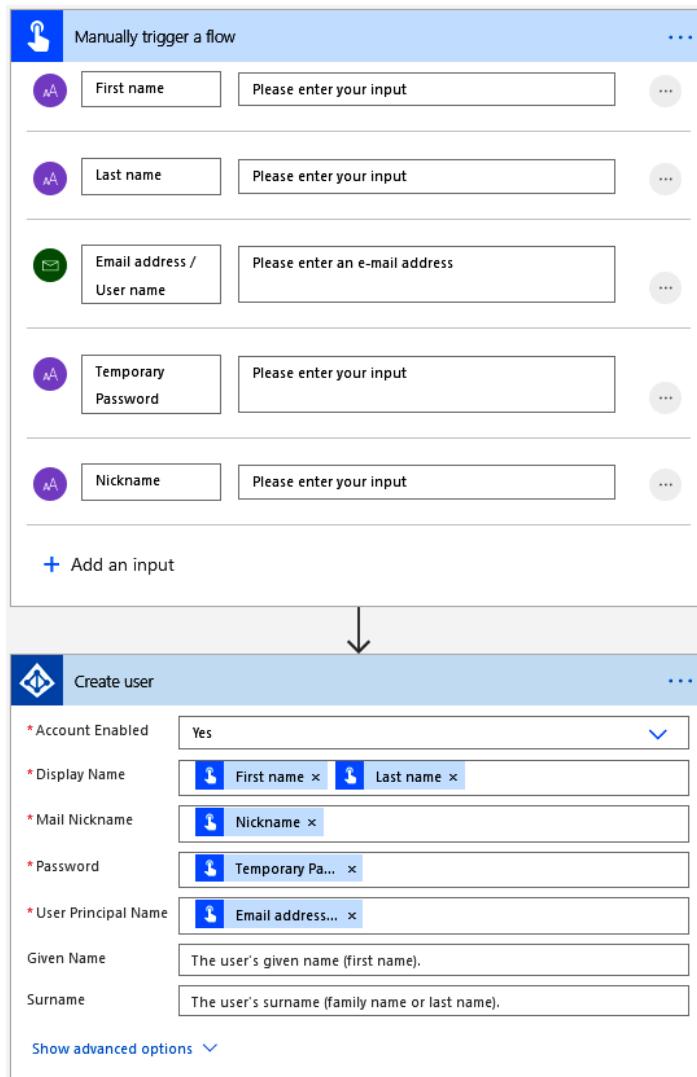


Figure 3.6 - Power Automate flow for creating an Azure Active Directory user

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 Active Directory 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 creation, apps creation, flows creation, connector usage, custom connector creation, 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 lifecycle management, the implementation approach and 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**
- **CDS auditing**

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

Office 365 Activity Logging

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

Office 365 Security & Compliance Center is a comprehensive tool used for managing security and compliance in the Microsoft SaaS 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. Office 365 Activity Logging needs to be enabled. 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 apps permission changes).
- **Power Automate** events (creating, editing, and deleting flows, as well as flows permission changes).
- **Model-driven apps** and **Dynamics 365** events (administration events, creating, reading, updating, and deleting data in CDS entities, 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 & 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 a 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 database.

Next, we'll look at CDS auditing.

CDS auditing

Another auditing option is to use traditional CDS auditing, which runs within a model-driven application and stores the auditing data in a specific auditing entity in the CDS database. Auditing only works within the boundaries of a CDS database and the amount of information that's tracked is smaller; for example, no read transactions can be audited.

Understanding application monitoring

Another important monitoring option is to monitor the Power Platform solutions 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 monitoring into the solutions is to make use of Azure Monitor's capabilities; that is, **Azure Application Insights** and **Azure Log Analytics**.

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 with server-side registered **Webhook** event handlers. Alternatively PlugIn error handlers can be used for any server-side events.

Canvas apps integration with Application Insights can be easily implemented by only configuring the Application Insights **instrumentation key** in the canvas apps 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 Power Platform administration and monitoring aspects. We also learned about administration centers, the Power BI admin center, and API administration. We also gained knowledge about administration and monitoring using *Power Automate* and *Azure DevOps*.

In the next section, we will learn about 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, a new customer purchases a Microsoft cloud service and obtains their tenant as part of their cloud subscription. When purchasing other Microsoft cloud services, they are added to the tenant, which is 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, combined with some of the following:

- Separate development and testing active directory environments are required.
- Customer has a complex enterprise ecosystem with a very heterogenous active directory structure.
- 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:

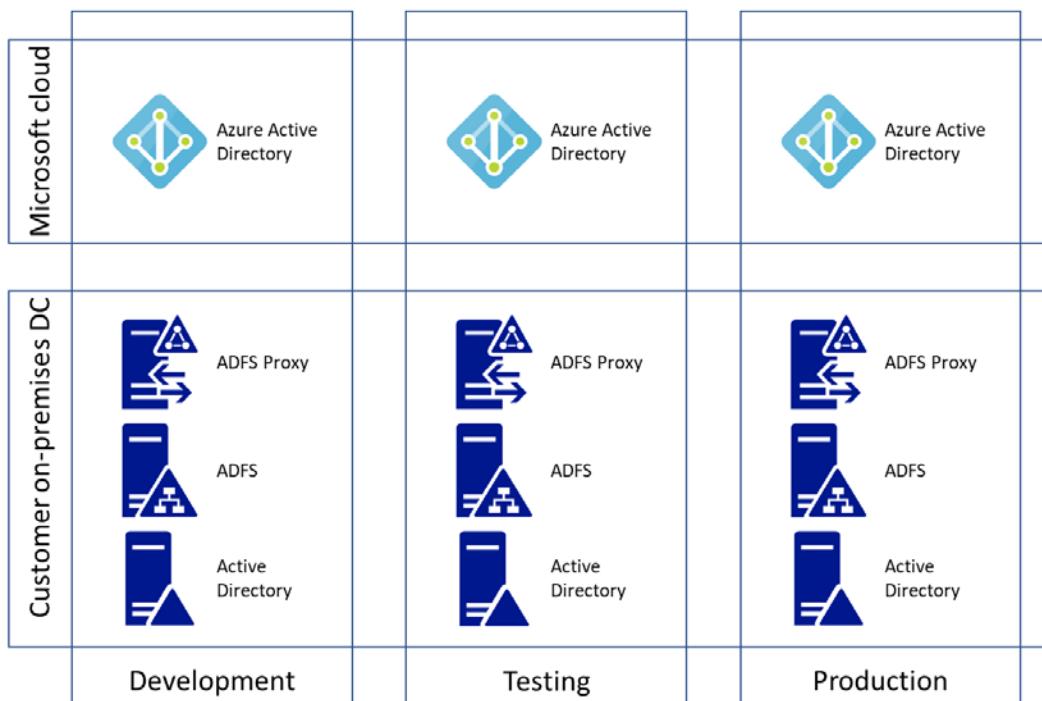


Figure 3.7 - Multi-tenant environment

When using this multi-tenant approach, any cloud services such as Microsoft Power Platform are usually deployed only in the production tenant; they do not impose any potential conflict with having identical cloud services in different tenants with separated and non-integrable sets of user credentials.

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, and so on.

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:

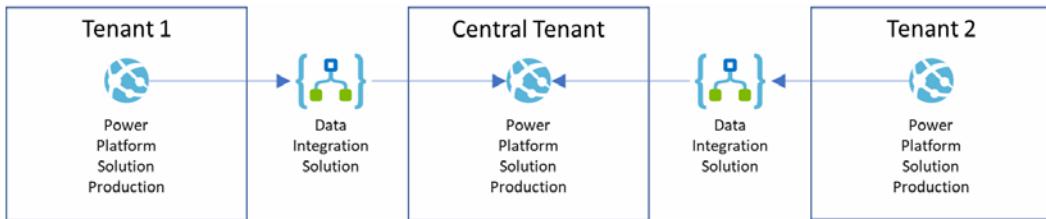


Figure 3.8 - Central consolidation environment

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:

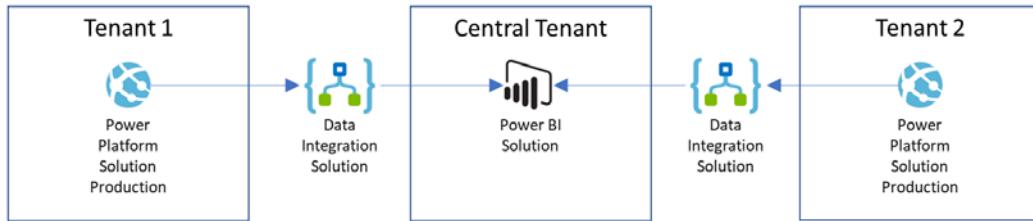


Figure 3.9 - Central reporting environment

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.

This second solution approach is beneficial for the typical situation that's encountered in large organizations, where headquarter management just needs analytics and reporting capabilities on the global data coming from the various organizational units.

Understanding environment setup

Designing an environment composition 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 two main purposes:

- Developing a Power Platform solution
- Operating a Power Platform solution in production

Based on all the various environment types we've looked at, you should know that only the following two types can be used for an enterprise solution's development and operation:

- 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 composition that is not labeled *Production* is of the *Sandbox* type.

These best practices are generally valid for any Power Platform solution, but specifically for CDS-based solutions. This is because, with the CDS database, another level of complexity must be considered.

Development environments

A **development environment** ecosystem is used to develop the main Power Platform-based solution. The solution's complexity, the complexity of the customer's IT ecosystem, and the duration of the project all influence the development environment's composition. In this section, we will present the following typical environment compositions:

- Composition for a **simple and short development**, which is used for small projects with very limited custom development parts.
- Composition for a **complex single-stream development**, which is used for longer projects with significant custom development efforts.
- Composition for a **complex multi-stream development**, which is used for long and complex projects, with the need to develop the solution using multiple parallel work streams.
- Composition for **complex testing requirements**, which might be necessary if the customer requires several testing stages using a cascade of testing environments.

Simple development

The simplest and most minimal required environment setup for developing a Power Platform solution consists of three environments:

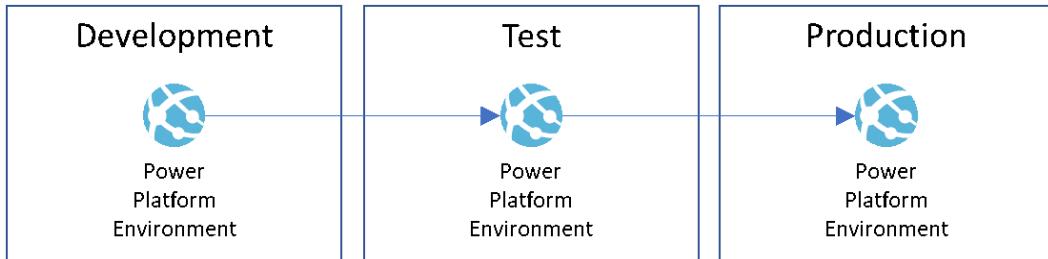


Figure 3.10 - Environment setup for simple development

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.

This setup is suitable for simple projects with small development teams and little to no custom development. Larger development teams would need a separate environment so that they don't interfere with each other. For extensive custom development, it is even more important to assign separate environments to developers, as described in the next composition.

Development team and a single workstream

In the case of a large and complex Power Platform solution with extensive custom development but with a single workstream, the setup needs to consider this complexity, as illustrated in the following example diagram:

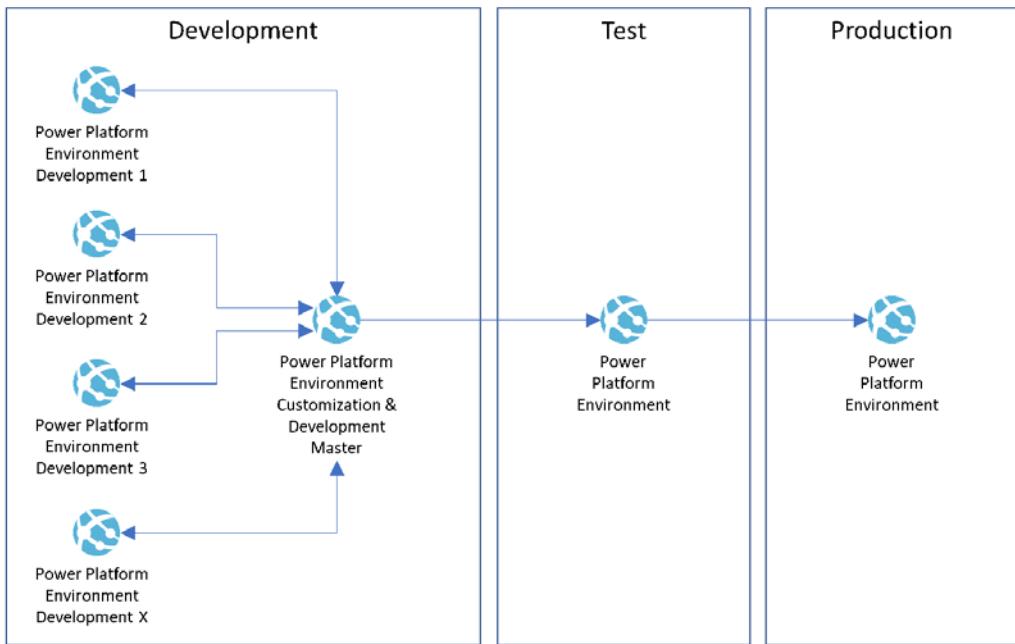


Figure 3.11 - Environment setup for large and complex single-stream development

The environments in the development phase will have the following roles:

- **Development (1, 2, 3, ... X):** These are separate environments for individual developers that customize well separated parts of the solution or develop custom developed artifacts.
- **Customization and Development Master:** This is an environment where all the customization for the CDS database could be performed centrally, and then deployed onto the development environments to keep them in their current state. This environment can also help with packaging solutions and preparing to export them into the subsequent environments.

Development team and multiple workstreams

For certain complex Power Platform projects, there is a need to split the project work into multiple streams in order to allocate domain experts to project groups. This can be the case when, for example, several Dynamics 365 workloads, such as Sales, Marketing, and Customer Service, are implemented in the scope of a large project. To reflect this project setup, the environment composition from the previous example needs to be replicated to allow developing in independent streams, but still giving the user the possibility to merge the final solution into one package, as illustrated in the following example diagram:

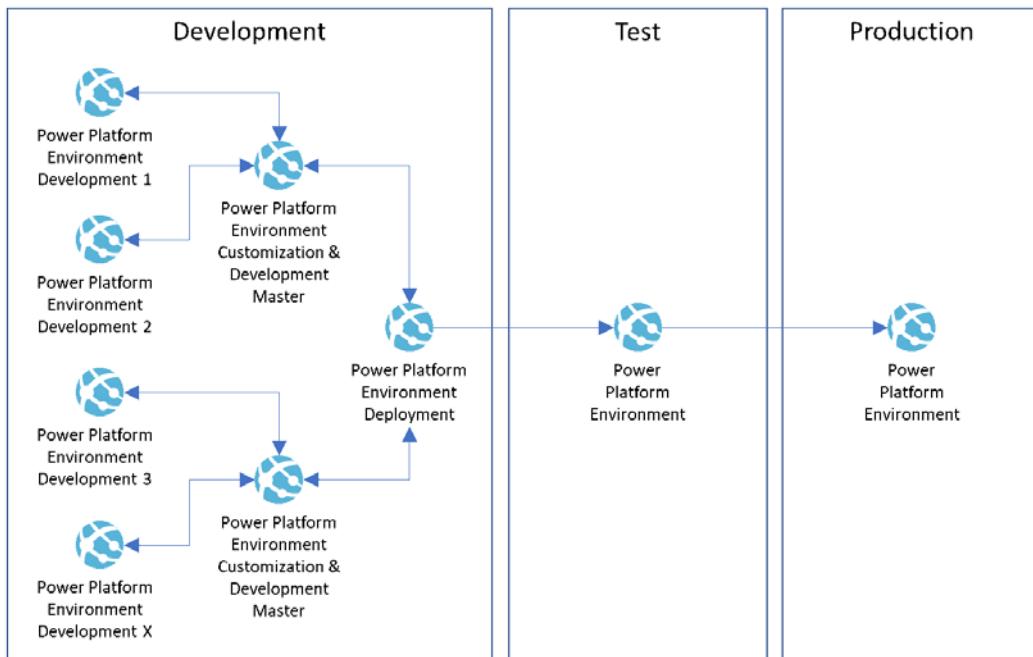


Figure 3.12 - Environment setup for large and complex multiple-stream development

The preceding diagram presents a composition of environments for two project streams. In each stream, there are environments for individual developers and a development master for consolidation within the stream. The **Deployment** environment is the ultimate consolidation layer as it brings all the results of all the streams together and prepares the consolidated solution for deployment into the testing cascade.

Complex testing

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

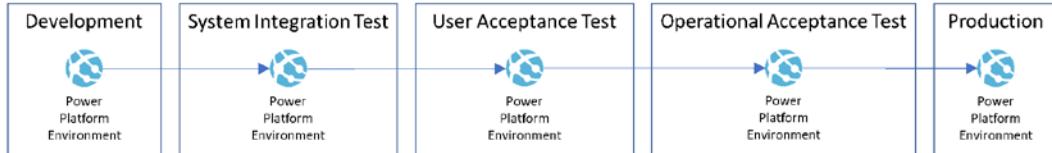


Figure 3.13 - Environment setup for complex testing scenarios

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 is as follows:

- **System integration testing (SIT):** This environment is used for technical testing of the solution as a whole, including all possible integrations with other IT systems.
- **User acceptance testing (UAT):** This environment is used for final end-to-end functional testing provided by human testers.
- **Operational acceptance testing (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.

Production environments

A production environment ecosystem is used after the main Power Platform-based solution has been finalized and brought into production.

Simple production

In the case of a simple solution with one single release, the simplest environment setup consists of development, testing, and production. This is illustrated in the following diagram:

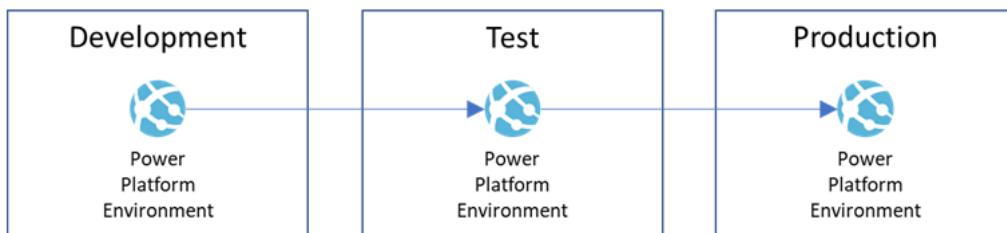


Figure 3.14 - Environment setup for simple production

The existing development and testing environments need to stay in operation for bug fixing and making small improvements.

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:

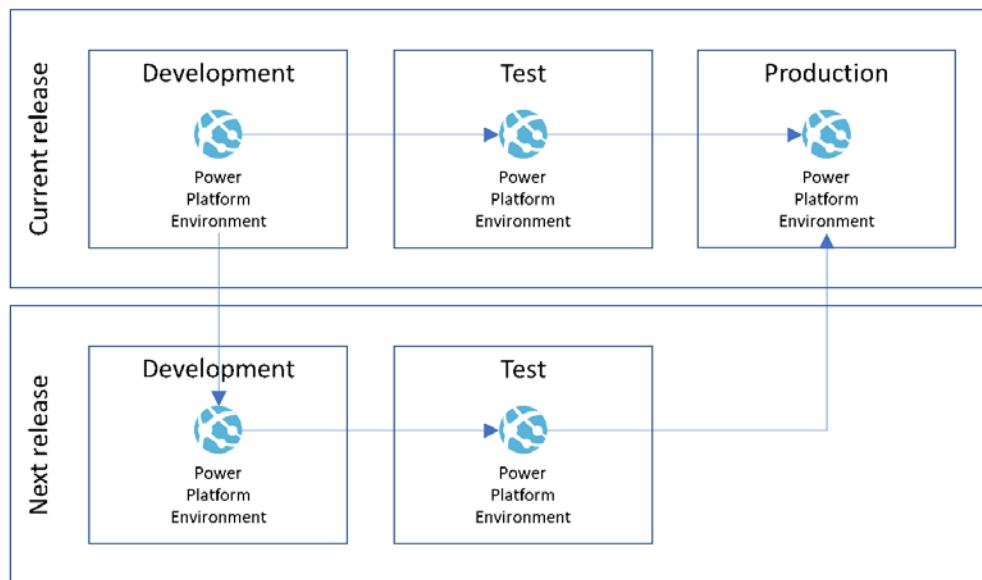


Figure 3.15 - Environment setup for the multiple release strategy

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 that will be released.

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 ensure 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 solution.

Product upgrades

In case of a longer-running solution implementation, where a new Power Platform product release is expected during the implementation, it is necessary to ensure 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 this purpose, for those dedicated environment(s), the preview feature must be activated at the beginning of the preview window (between the preview's availability and new product's release date) in order to ensure enough time for all necessary tests.

This setup is shown in the following example diagram:

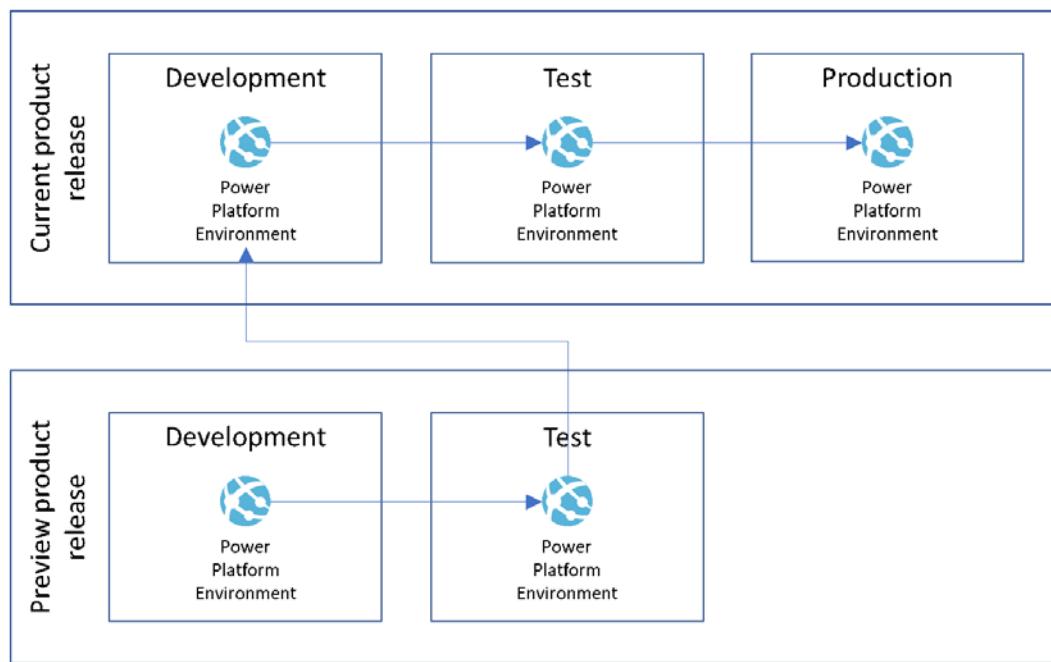


Figure 3.16 - Environment setup for product upgrades testing

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. This way, they can consider the issues that have been identified in the main development environment.

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 the AppSource or other sources to decide on the feasibility of the developed solution. This 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
- Performance requirements

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. There are the following risks:

- Governance over environment creation
- Security risks connected with *shadow IT*
- 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, some of which were already described earlier in this chapter, such as proper use of DLP policies or using monitoring tools. In order to support customers so that they can achieve 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 CDS solutions, model-driven apps, canvas apps, Power Automate flows, connectors, Power BI dashboards, 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 auditing
- 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:

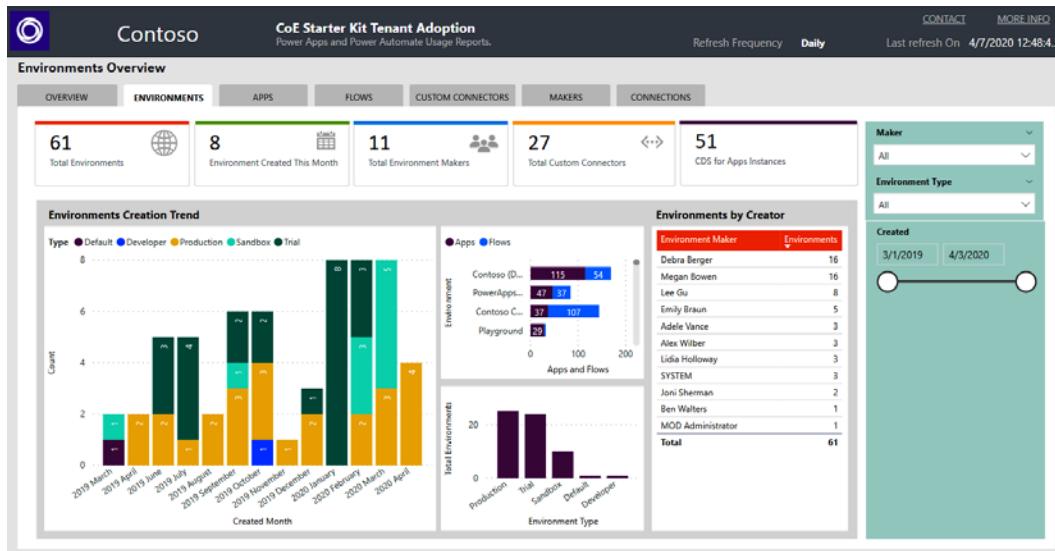


Figure 3.17 - CoE Starter Kit – Environments Overview

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.

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:

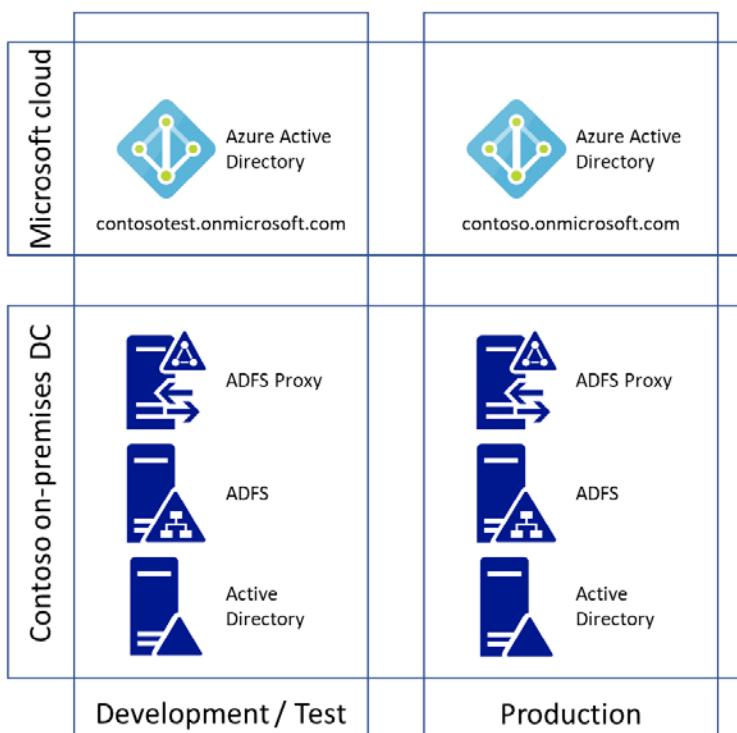


Figure 3.18 - Contoso Inc. tenant composition

As documented in the tenant structure, there will be a development and test tenant. This will be used to develop an integration between their on-premises active directory and the cloud Azure Active Directory. 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; that is, contoso.onmicrosoft.com.

It was further decided that there will be a single Power Platform **production environment** for world-wide 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:

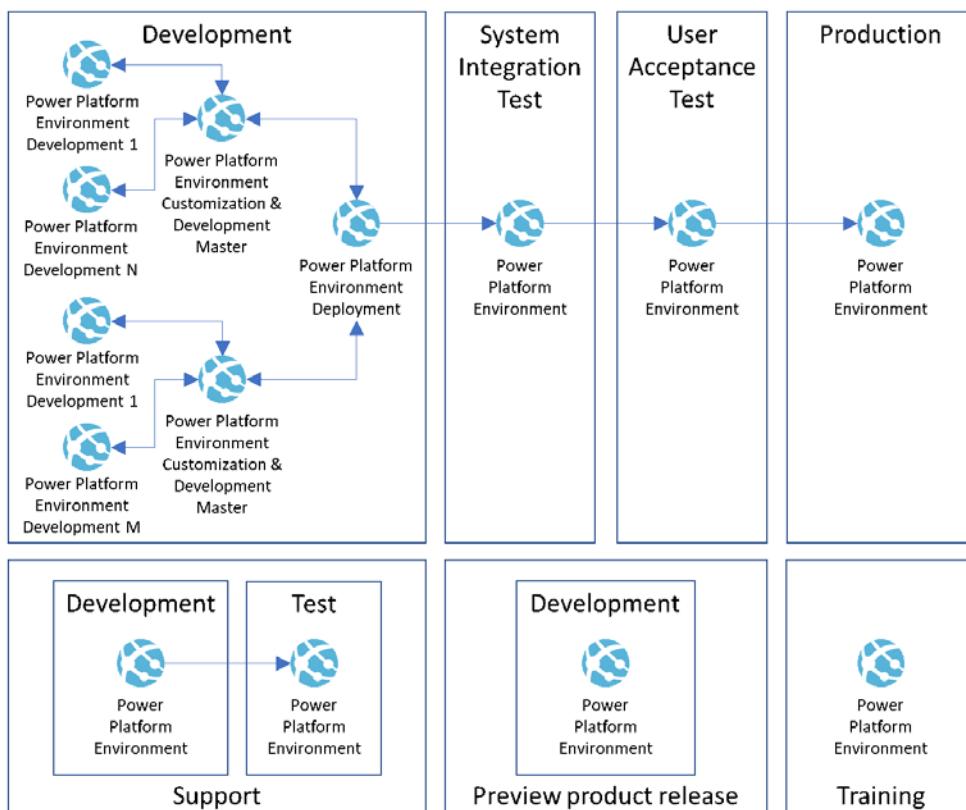


Figure 3.19 - Contoso Inc. environment configuration

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 a next release is being developed on the primary development environments.
- **Preview product release:** This will be an environment where the product preview will be always 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, which 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 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*, including its 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 *CDS 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

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 using of various of 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 to understand the different toolsets used by these two categories of developers.

Introducing tools and techniques

Before we start introducing the various tools that are used for Power Platform solutions 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 was 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.

Introducing the IT pro developer

The **IT pro developer** is a traditional role in the industry. IT pro developers usually have 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 in 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, development tools, and simple formulas*
- **IT pro developer:** Performs custom development of complex solutions using real *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 rather an additional option to open the area of building apps to 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 for custom development.

Presenting configuration and customization tools

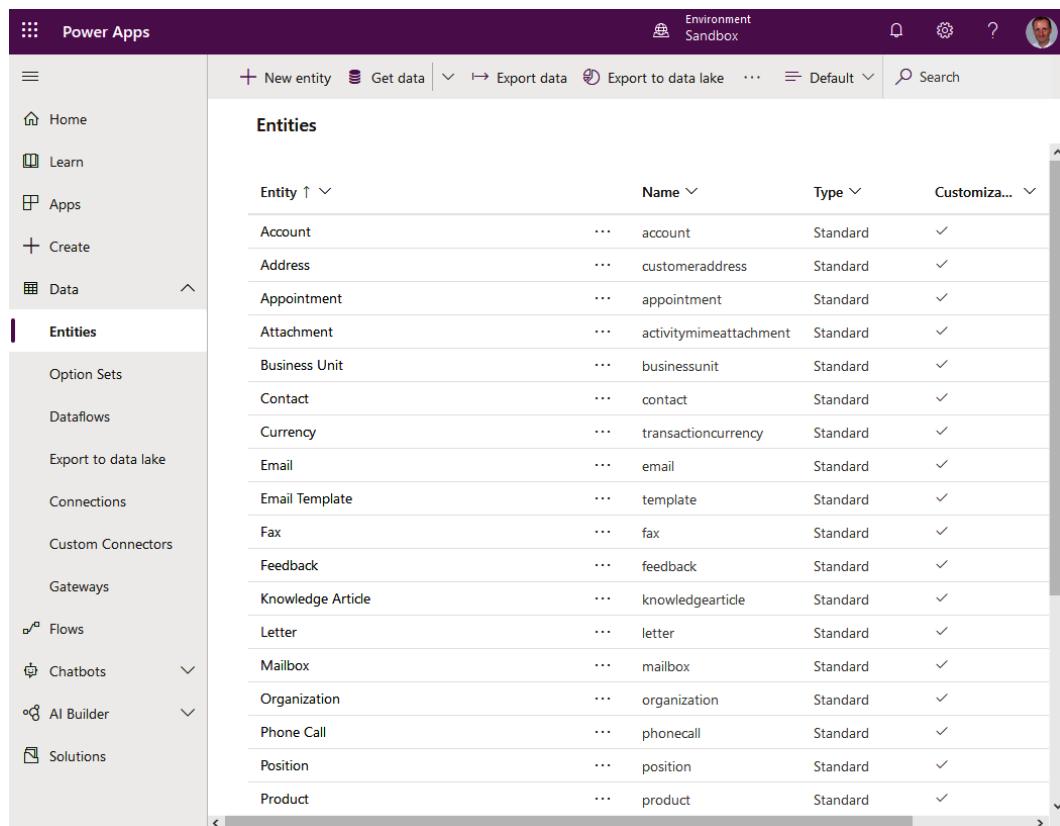
In this section, you will learn about the most important Microsoft and community tools that are used primarily for configuring and customizing Power Platform solutions. This toolset can be used by citizen developers, as well as IT pro developers.

Knowing Common Data Service (CDS) and model-driven apps 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 **CDS solution development**.

Introducing Maker Portal

The main customization tool for building CDS and model-driven apps is the **Maker Portal**. You can access Maker 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. The left sidebar has a navigation menu with sections like Home, Learn, Apps, Create, Data, Entities (which is currently selected), Option Sets, Dataflows, Export to data lake, Connections, Custom Connectors, Gateways, Flows, Chatbots, AI Builder, and Solutions. The main content area is titled "Entities" and lists various entities with columns for Entity, Name, Type, and Customiza... (with a checkmark). The listed entities include Account, Address, Appointment, Attachment, Business Unit, Contact, Currency, Email, Email Template, Fax, Feedback, Knowledge Article, Letter, Mailbox, Organization, Phone Call, Position, and Product.

Entity ↑	Name	Type	Customiza...
Account	account	Standard	✓
Address	customeraddress	Standard	✓
Appointment	appointment	Standard	✓
Attachment	activitymimeattachment	Standard	✓
Business Unit	businessunit	Standard	✓
Contact	contact	Standard	✓
Currency	transactioncurrency	Standard	✓
Email	email	Standard	✓
Email Template	template	Standard	✓
Fax	fax	Standard	✓
Feedback	feedback	Standard	✓
Knowledge Article	knowledgearticle	Standard	✓
Letter	letter	Standard	✓
Mailbox	mailbox	Standard	✓
Organization	organization	Standard	✓
Phone Call	phonecall	Standard	✓
Position	position	Standard	✓
Product	product	Standard	✓

Figure 4.1 - Power Apps Maker Portal

In Maker Portal, the user can do the following:

- Customize the CDS database and create or modify entities, fields, relationships, forms, views, charts, and dashboards
- Customize global option sets
- Trigger the separate designers for model-driven apps, canvas apps, PowerApps portals, Power Automate flows, and Power Query dataflows
- Configure the synchronization of CDS data in Azure Data Lake

- Manage connections and custom connectors
- Manage on-premises data gateways
- Work with the Power Virtual Agent 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, 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.

Introducing the model-driven app designer

After the CDS 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 CDS 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 CDS. This helps avoid confusion and increases user adoption.

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

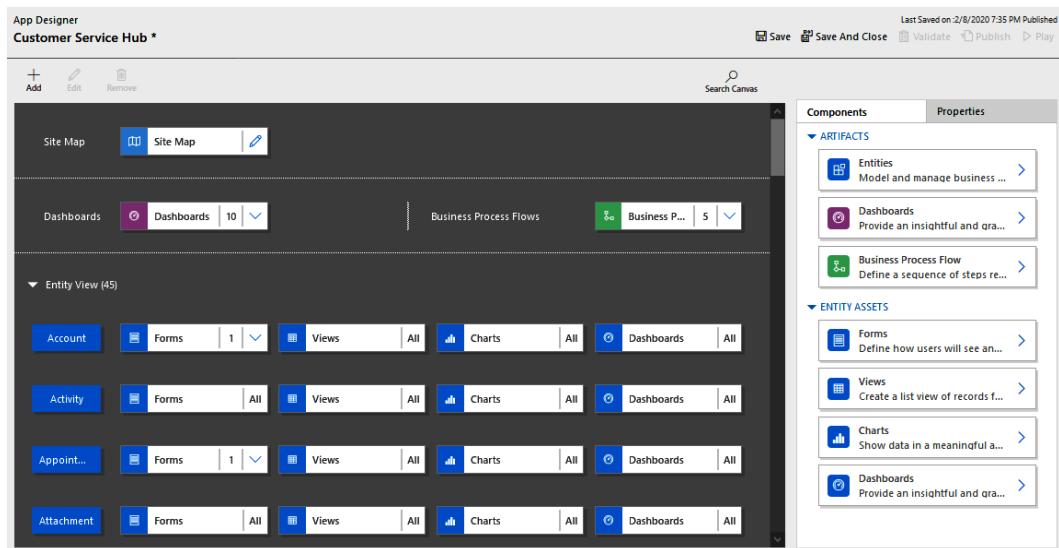


Figure 4.2 - Model-driven app designer

The designer provides the following capabilities:

- You can create and modify a model-driven app by selecting entity-independent dashboards, business process flows, as well as entities for the model-driven app. For every entity, you can select the UI components, such as forms, views, charts, and entity dashboards to be included in the model-driven app.
- You can trigger the **Site Map designer** in order to configure the app's navigation.

Next, we will describe the capabilities of the site map designer for model-driven apps.

Introducing the model-driven app site map designer

Every model-driven app needs to have configured the main navigation of the app, called the **site map**. The site map defines navigational areas and in each navigational area, there are groups of navigation elements called **subareas**.

Sitemap Designer, as illustrated in the following screenshot, can be triggered directly from the app designer:

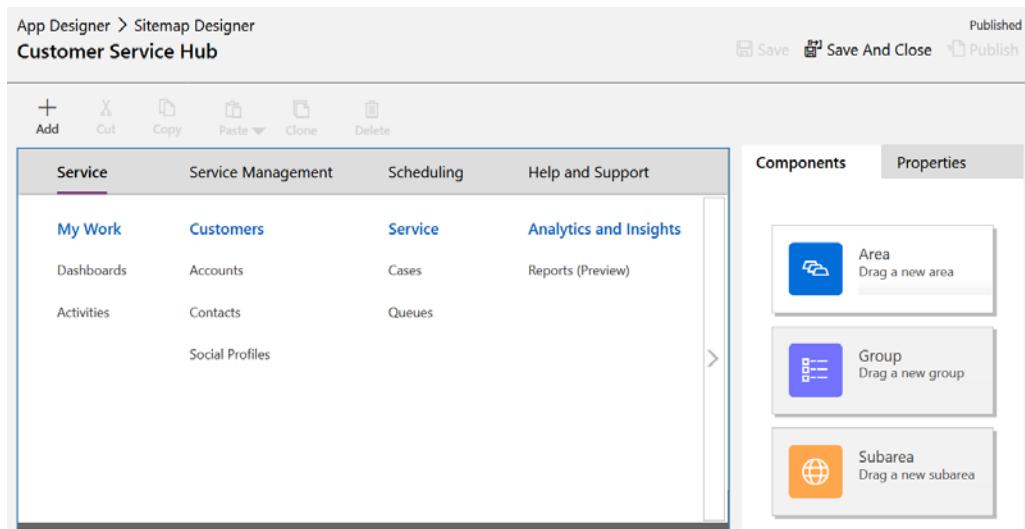


Figure 4.3 - Model-driven apps Sitemap Designer

The designer is used to create and modify the model-driven app's three-level navigation:

- **Area:** Represents the navigation areas selectable in model-driven apps
- **Group:** Represents a group of navigation elements within an area
- **Subarea:** Represents the individual navigation elements (entity views, dashboards, or other elements)

Besides those main configuration and customization tools, there is a multitude of third-party tools that make the life of a CDS 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, it is specifically the **Ribbon Workbench** that has gained a large amount popularity and general adoption in the expert community.

Ribbon Workbench

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

Many other tools from this toolbox are filling gaps in 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.

Introducing PowerApps Portal Studio

The main customization tool for building PowerApps portals is **Portal Studio**, which can be triggered directly from the *Maker Portal*. Portal Studio is illustrated in the following screenshot:

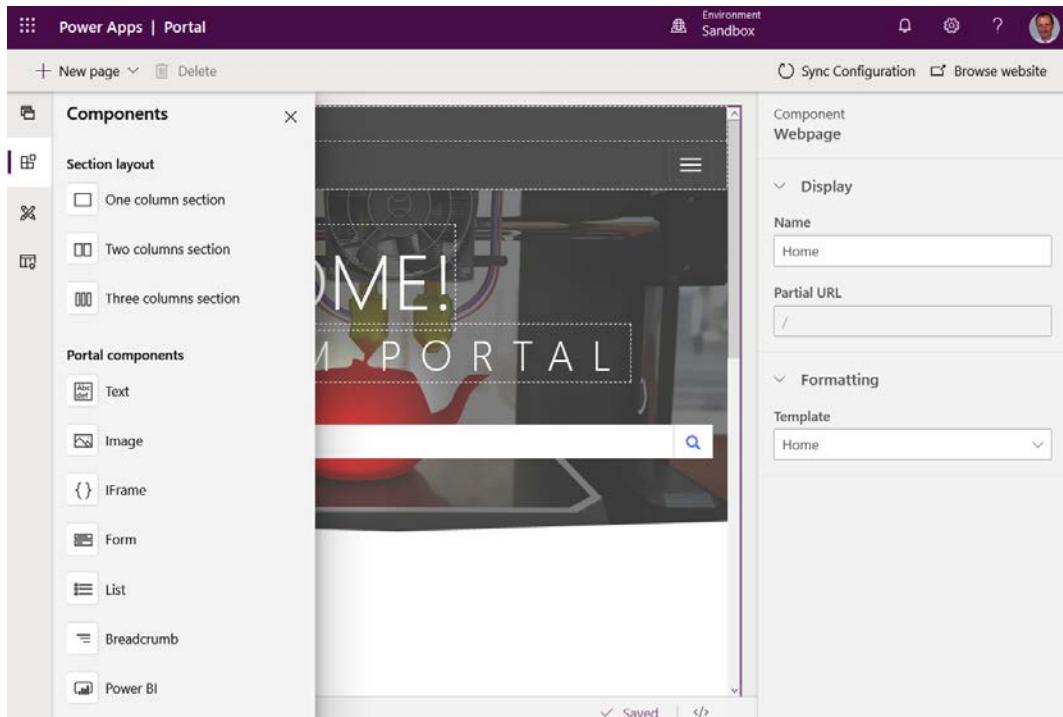


Figure 4.4 - Power Apps Portal Studio

Portal Studio has the following capabilities:

- Creating and modifying portal pages using page templates
- Using various page components from the components gallery
- Theming the customized portal
- Configuring the portal pages using a *graphical UI* or code

Web development experts can develop parts of the portal solution using *HTML*, *JavaScript*, and other technologies, programming and script languages.

Introducing canvas apps Designer Studio

The main customization tool for building canvas apps is the **Designer Studio**, which can be triggered directly from Maker Portal by using the following URL (this depends on the region of the environment – this example link is for Europe): <https://eu.create.powerapps.com/studio>.

Designer Studio is illustrated in the following screenshot:

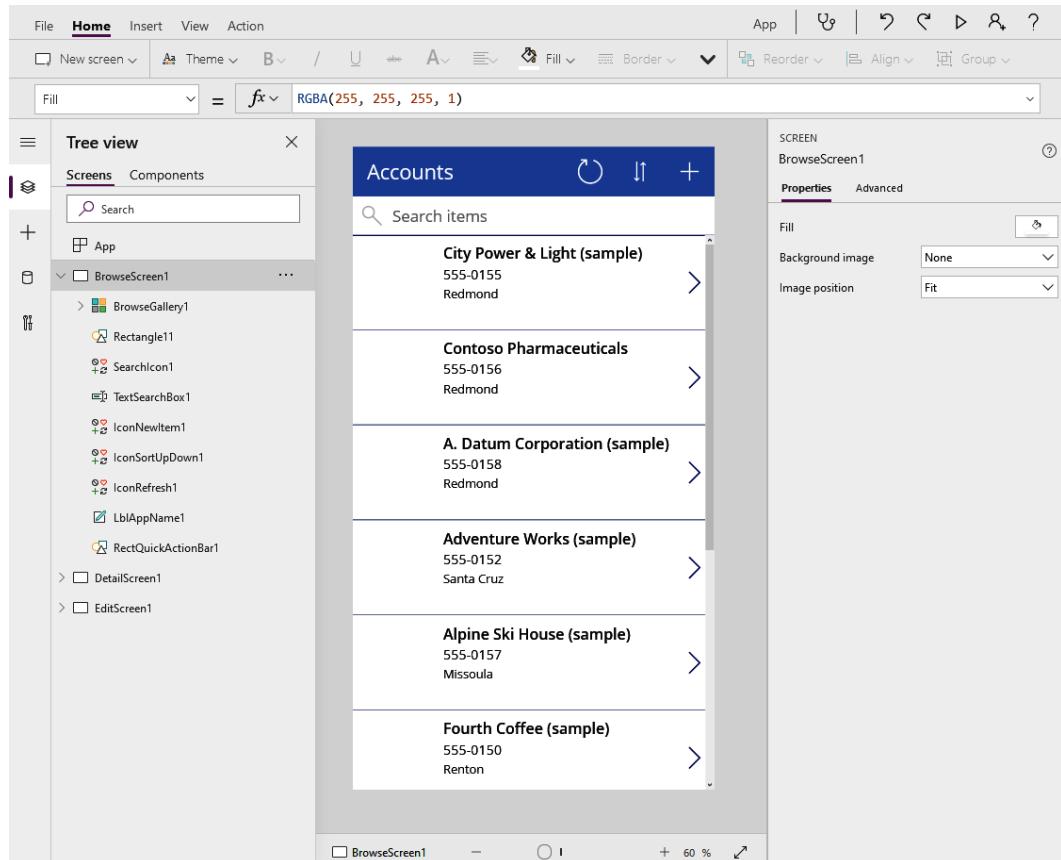


Figure 4.5 - Canvas apps Designer Studio

Designer Studio provides the following capabilities:

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

In the following section, we will focus on the Power Automate designer.

Introducing the Power Automate designer

The main customization tool for building Power Automate flows is the **Power Automate designer**, which can be triggered directly from the Maker Portal by going to the following URL (this URL depends on the region of the environment): <https://emea.flow.microsoft.com>.

The designer is shown in following screenshot:

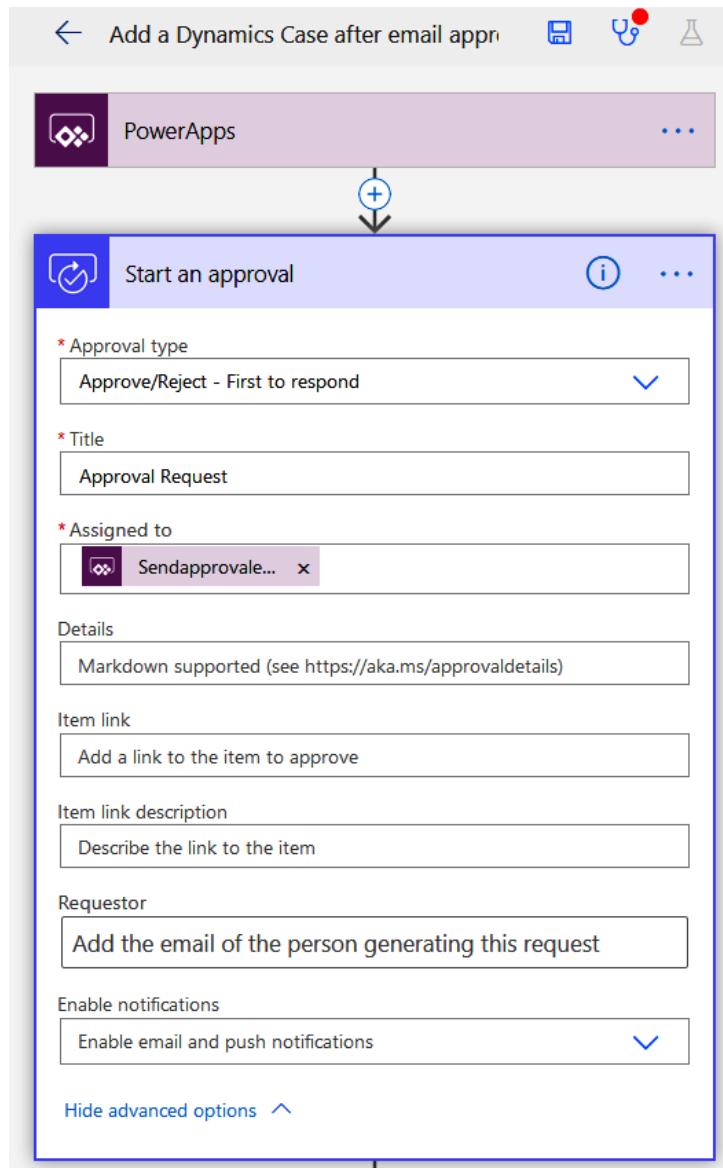


Figure 4.6 - Power Automate designer

The designer 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 the flows, triggering button flows, and managing approval flows.

Introducing the Power Virtual Agents designer

The **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 the PVA designer, you can do the following:

- 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 canvas graphical designer environment. The flow of the dialog can contain questions, messages, branching, and other logic.
- Create entities representing certain typical information units, which helps 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 into several technologies, to serve the purpose as automated first-level support. PVA bots can be integrated with any websites, Microsoft Teams, Facebook, or mobile applications. Part of the PVA designer is also built-in analytics so that you can analyze the usage and performance of your bots.

Introducing 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 is part of 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 CDS solutions, integrated with canvas apps, or used in Power Automate flows.

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

- Prediction
- Category classification
- Entity extraction
- Object detection
- Form processing

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

- Business card reader
- Text recognition
- Category classification
- Entity extraction
- Key phrase extraction
- Language detection
- 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.

Introducing the dataflows designer

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

- You can select a data source from a broad range of possible data sources covering all major database systems, as well as many other popular technologies.
- You can transform and filter the source data using various transformation and filtering options.
- You can load the data into an existing or new CDS entity, as well as perform an appropriate field mapping for the data load.
- The data load can be a one-time process 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.

Getting to know 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 Maker Portal.

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.

Learning about 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:

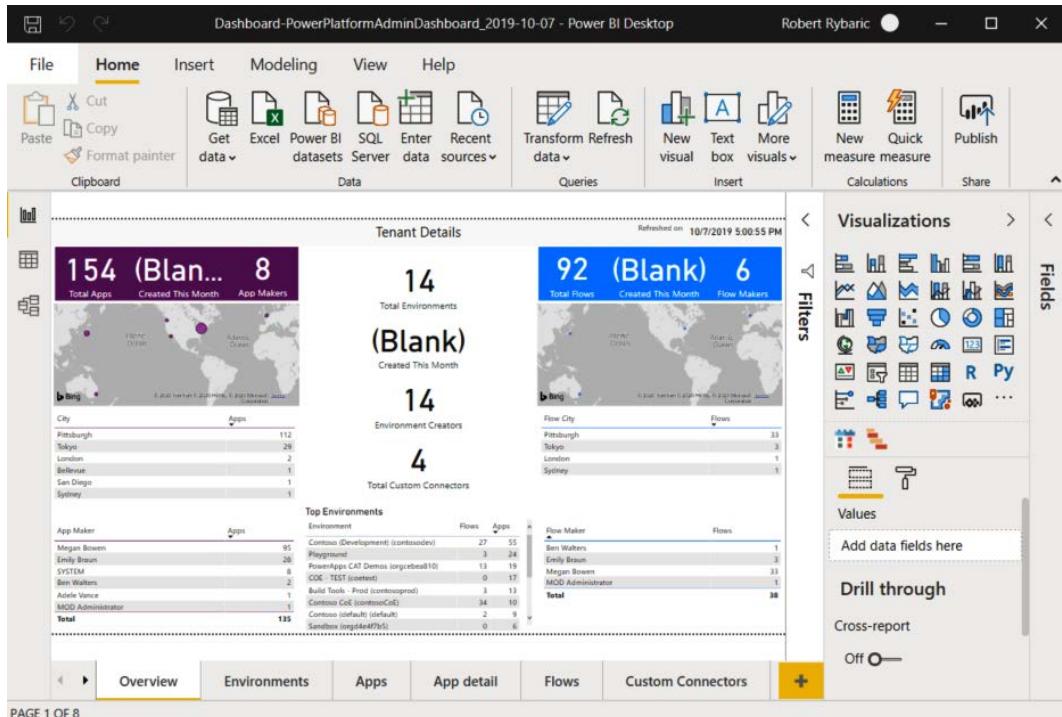


Figure 4.7 - Power BI Desktop

Power BI Desktop has the following capabilities:

- You can connect to multiple different data sources, transform the data, and create a data model.
- You can create the required visualizations on top of the data model.
- You can create reports.
- You can publish reports.

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

Learning about Power BI Builder

Power BI Builder is a designer tool that's used for building paginated reports. Paginated reports are the opposite of Power BI reports, and dashboards 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 dataset**
- **Premium Power BI dataset**
- **Enter data**

Power BI Builder has the following capabilities:

- You can create paginated reports using the aforementioned *data source technologies*.
- You can publish 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.

Learning about Power BI Service

Power BI Service is the cloud service where Power BI content is deployed and runs. 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 designer 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 are the services provided by Power BI:

- Create dashboards, apps, and workspaces
- Create dataflows
- Share Power BI components with other users

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.

Understanding Dynamics 365 Customer Voice designer

Dynamics 365 Customer Voice is used to create surveys and can be found at <https://forms.office.com/FormsPro/Pages/DesignPage.aspx>.

The designer has the following capabilities:

- You can design and configure surveys with question types and business logic.
- You can distribute surveys using the available distribution channels, including email (with or without using a template), Power Automate, embedding, providing a link URL, or using a *QR code*.
- You can evaluate responses using built-in analytics.

It is possible to integrate surveys created with the *Dynamics 365 Customer Voice* with other Power Platform solution components, specifically with Dynamics 365 applications with Power Automate flows or other integration approaches.

Understanding 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.

Understanding ISV Studio

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

In this section, you learned about the various tools that are used for configuring and customizing Power Platform components. Depending on your planned Power Platform-based solution, you will need to use some or many of these 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 primarily focuses on IT pro developers.

Learning about 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 the 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:

- CDS plugins
- CDS custom workflow actions

- Listeners for Azure Service Bus integration scenarios
- Any CDS 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*.

Learning about 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 the 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)**. **PowerApps 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 the CDS frontend web resources or some Azure solutions, can be developed with Visual Studio Code.

Introducing the Power Apps CLI

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

Introducing Power Platform extensions for Visual Studio

On the Visual Studio Marketplace, there are several free as well as commercial extensions for Power Platform development. Most of these extensions support the development of CDS and Dynamics 365 code artifacts – plugins, custom workflow actions, and web resources – by providing ready-made solution 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 the 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 save unneeded repetitive work.

Learning about 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 PowerApps (<https://docs.microsoft.com/en-us/dynamics365/customerengagement/on-premises/developer/download-tools-nuget>).

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.

Introducing the code generation tool

This tool is used to generate classes for early-bound programming against the **Organization Service (SOAP)** CDS API. 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 CDS metadata. These programming styles will be discussed in detail in the *Chapter 8, Microsoft Power Platform Extensibility*.

Introducing the plugin registration tool

This is the main developer tool for registering several different development artifacts, including plugins and custom workflow actions, as well as several cloud connections such as Azure Service Bus, Azure Event Hub, and webhooks, with the CDS solution. The registered artifacts or connections serve then as event handlers propagating CDS data to the recipients.

Introducing 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 plugin management, enhanced early-bound class generation, *FetchXml* building, enhanced *solution management*, browsing *CDS metadata*, enhanced support for building *PCF controls*, enhanced *PowerApps Portals* management, and enhanced web resources management.

Introducing Postman

Postman is a third-party tool used for developing and testing applications that are communicating 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:

- You can create development and testing environments and save configurations.
- You can support a broad range of authentication scenarios, ranging from no authentication, basic authentication, and *OAuth* to certain vendor-specific authentication types.
- You can easily build requests, execute them, and analyze the responses.

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

Introducing CRM REST Builder

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

Introducing 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 many more with a predefined set of test packages.

Testing backend components

Testing backend components for CDS applications (plugins, custom workflow actions, and so on) is not straightforward, since these components are deployed in the CDS database and are running in the cloud, where a developer cannot establish a connection and debug the components. There are various approaches for testing these components, such as using *tracing* or using the *plugin profiler*. However, you can use an alternative approach and test locally using mocked or faked data by leveraging some of the supporting frameworks. The 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/>)

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 specific requirements for business functionalities and the user interface, or with complex integration and data migration requirements, custom development might be necessary. In this section, you have learned about the most important custom development tools and now you should have a good understanding and be able to adopt those tools for 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.

Introducing 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 the Power Platform solutions deployment. In the following section, you will learn about the capabilities of these three tools.

Configuration migration tool

Configuration migration tool (<https://www.nuget.org/packages/Microsoft.CrmSdk.XrmTooling.ConfigurationMigration.Wpf>) is used to create a **data migration package** to migrate smaller data amounts from one CDS environment into another. It is used preferably for migrating static configuration data from several entities in one package. The advantage of this tool is that it combines data from many entities into a single package while preserving relationships between records in the entities.

Package deployer tool

Package deployer tool (<https://www.nuget.org/packages/Microsoft.CrmSdk.XrmTooling.PackageDeployment.Wpf>) 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.

Solution packager tool

Solution packager tool (<https://www.nuget.org/packages/Microsoft.CrmSdk.CoreTools>) 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.

Introducing 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.

The tool's appearance is illustrated on the following screenshot:

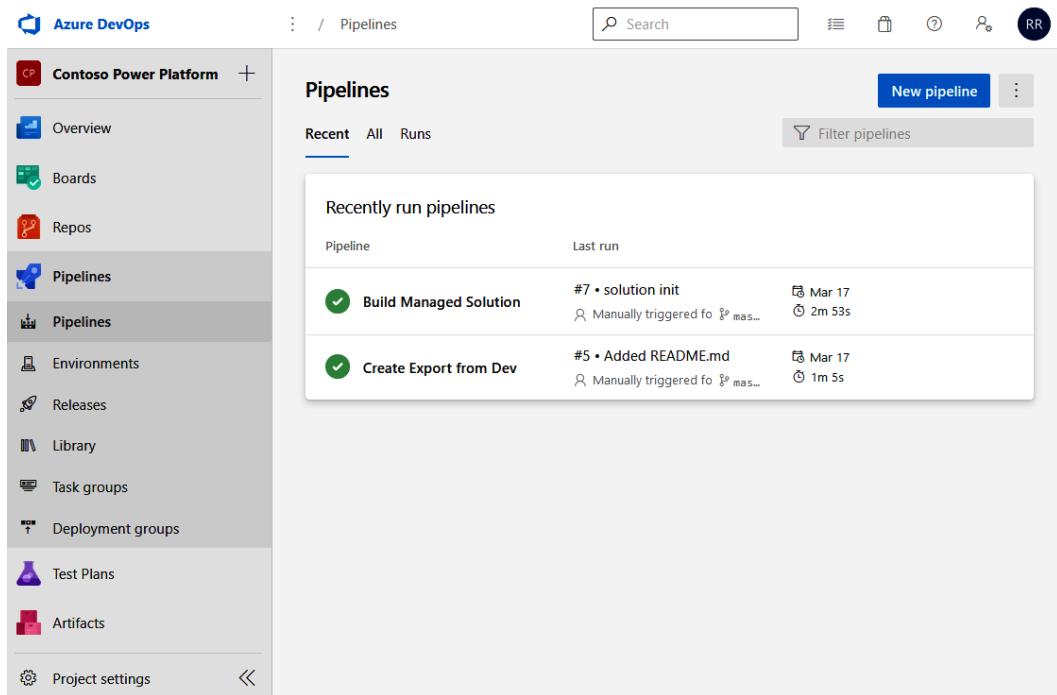


Figure 4.8 - Azure DevOps

Azure DevOps provides the following main capabilities:

- Agile planning and work tracking tools in **Azure Board**
- Repositories for source control in **Azure Repos**
- Build and release automation tools in **Azure Pipelines**
- Testing support with **Azure Test Plans**
- **Azure 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 Apps checker
- **Helper tasks** to support the installation of Power Apps tools

Important note

You will dive much deeper into the ALM details when you use *Azure DevOps* in *Chapter 5, Application Lifecycle Management*.

Contoso Inc. project team workplace setup

The Contoso Inc. project management team has made themselves 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:

- CDS Maker Portal, along with all designers and studios
- Power BI service
- PVA designer
- Dynamics 365 Customer Voice designer

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

- Power BI Desktop
- Power BI 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. The **Power Apps 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*. The empowerment of citizen developers will be established gradually as the development of the Power Platform solution progresses and management collects deeper experiences.

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 next steps into the Power Platform.

In *Chapter 5, Application Lifecycle Management*, we will focus on the topic of 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 used to be very heterogeneous and complex, and proper ALM is the key to 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.

Understanding application lifecycle management

Application lifecycle management (ALM) is a crucial governance principle and approach for developing every complex 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, a 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.

Getting to know 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 heavy use of custom development, which will influence the deployment complexity.
- **Geographical coverage (local versus global):** The geographical coverage might require performance optimization as one of the key solution requirements.
- **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 in several languages.
- **Existing IT landscape and legacy systems:** The existing IT landscape might require the use of specific customizations compatible with the 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 experiences 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):** 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.

Understanding 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, specifically a complex one, consists of various different components. A complex Power Platform solution can consist of the following types of components:

- **Common Data Service (CDS) customization**
- CDS backend custom development
- CDS frontend custom development
- CDS configuration data
- Model-driven apps
- Canvas apps and canvas apps components

- Power Automate flows
- Custom connectors for canvas apps and Power Automate flows
- Power BI solution artifacts
- AI Builder models
- Power Virtual Agents, Power Apps Portals, and Dynamics 365 Customer Voice configurations
- Microsoft Azure solutions
- On-premise custom development artifacts

There are many other components, but all of them cannot 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 that 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 covered with the following main concepts and tools:

- Power Platform solutions management
- Azure DevOps with the Power Platform Build Tools

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.

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

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

Getting an 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 with the exception of Power BI. 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 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 of transporting solution components between environments.

Currently, there are the following solution components that 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 that can be included in a solution is growing constantly:

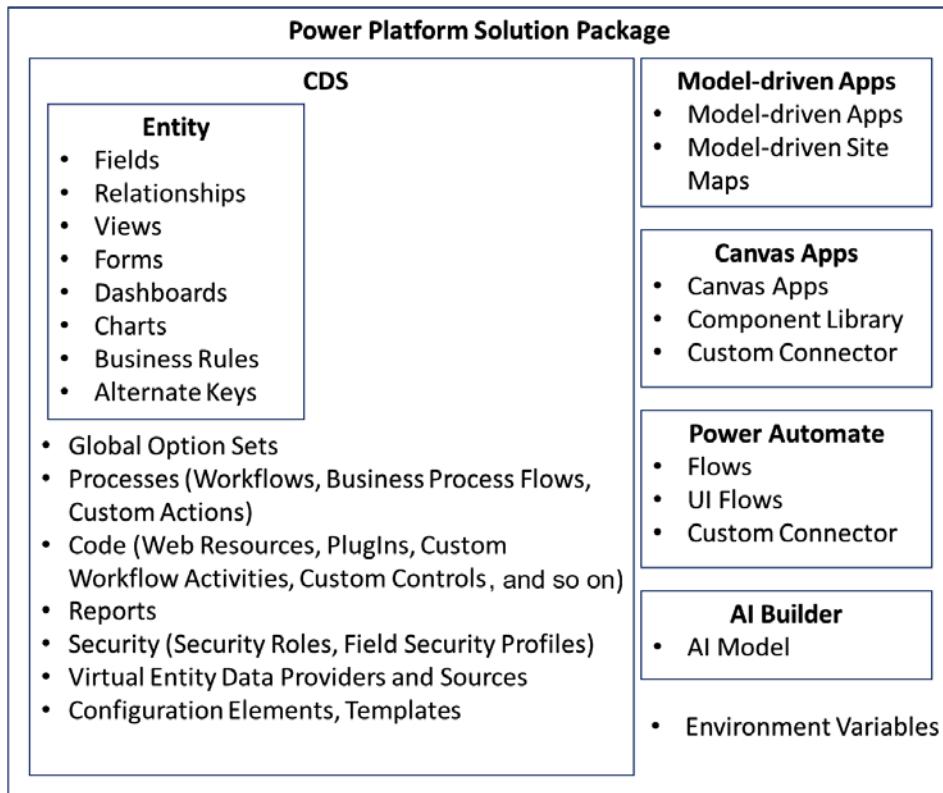


Figure 5.1 - Solution package structure

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 PowerApps Maker Portal or with code. A variable is managed within a specific *CDS entity* and can be accessed from any Power Platform solution component. A variable can be one of the following data types: **text**, **JSON**, **decimal number**, and **two options**.

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. Instead, any solution component can reference environment-specific information from the variables, such as the following:

- Environment URLs
- Registration values for *ISV* solutions
- Initial setup and configuration parameters used across apps and other settings

Moreover, the use of environment variables is fully supported in Azure DevOps.

Understanding 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 *CDS entity*, referenced in the solution package. A 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:** It is a text string that will be put in front of the technical name of each CDS artifact created within a solution referencing the solution publisher.
- **Option Value Prefix:** This is a five-digit autogenerated integer value between 10,000 and 99,999 that will be used as a prefix for all option sets 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 by dots, for example, **1.0.10.125**.

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, a 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.

Knowing about solution types

One of the key concepts of the Power Platform solution packages is the **solution type**. There are two basic solution types. We will learn more about these solutions and their features in the following sections.

Unmanaged solution

Every solution package when first created manually or with code is of the **unmanaged** type. Both 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 a managed solution.
- If an unmanaged solution is deleted from an environment, the contained components are not deleted; basically, they stay completely unchanged.

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.

Default solution

In every Power Platform environment, an unmanaged solution called a **default** solution is always automatically created, which is connected with an automatically created **default** publisher. The default solution always contains all solution artifacts contained in the Power Platform environment.

Learning about 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.

Knowing the properties of Forms, Views, Charts, and Dashboards

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

- Can be customized
- Can be deleted

Knowing the properties of Fields

For fields, the following managed properties are available:

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

Knowing the properties of entities

Entities have the most managed properties available:

- They 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.

Understanding dependencies and solution segmentation

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.

Introducing solution layering

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

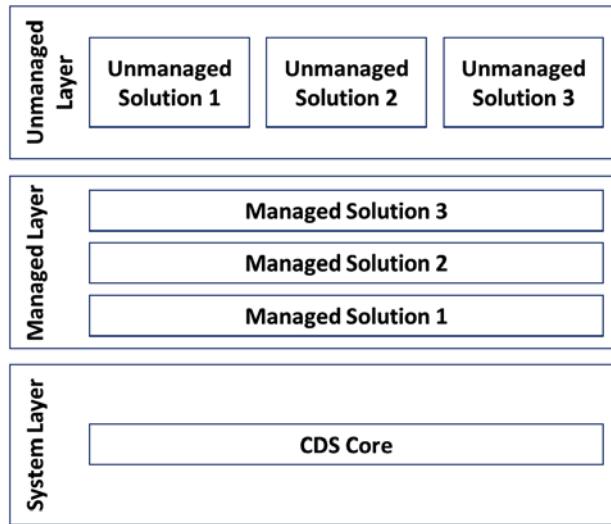


Figure 5.2 - Power Platform solution layering

The layers and their behavior are as follows:

- **System layer:** This contains the core *CDS entities*, and there is nothing a developer can do to influence this layer.
- **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 already installed managed solutions and may depend on their components. All purchased Microsoft first-party applications (*Dynamics 365 applications*) are always 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 layering behavior or, in other words, what is the result of having different customizations in several layers. You will learn about it in the next section.

Understanding layering behavior

The *solution layering* approach has a specific consequence on what is the result of customizations of the same component in the described layers. The customizations are applied, combined, and merged across all layers, bottom up. An individual component, which might be part of the core *CDS data model*, could be customized in some managed solutions and in the **unmanaged layer**, as illustrated in the following diagram:

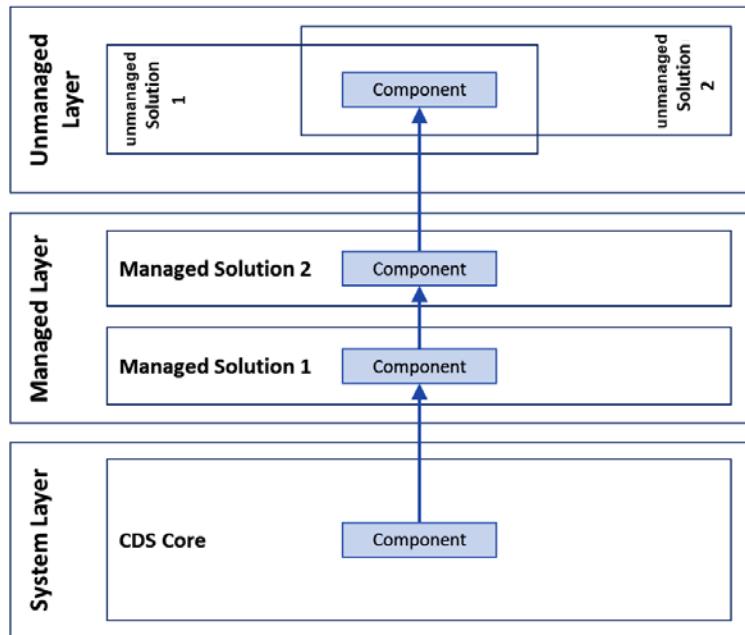


Figure 5.3 - Solutions layering behavior

As we can see, the ultimate customization of a certain solution artifact can be a combination or merge of several partial customizations. 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 in an environment.

Learning about 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** on an environment without having **Managed Solution 1** already imported.
- A solution cannot be deleted from an environment when any components of any solution above 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 *legacy Dynamics 365 customization tool*, there is a possibility to show dependencies even on the level of the individual solution components.

Learning about 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 makes it possible to include only a **subset** of all of the entity subcomponents in a solution package, not just the whole entity 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 and so significantly supports the adoption of the *ALM approach* within Power Platform solutions.

Introducing solution patches

Solution patches are small changes of 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 every time a full solution for even the smallest solution modifications.

Understanding solution updates

Solution updates are larger changes of some of the solution components usually used to provide some new functionality and, at the same time, to 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 update 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 case some component from a previous solution version is no longer needed and we would like to remove it. 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:

The screenshot shows a user interface for 'Import Options'. At the top left is a 'Import Options' button and a 'Help' icon. A yellow warning bar at the top right says: '⚠ This solution package contains an update for a solution that is already installed.' Below this is a section titled 'Solution Action ([Learn more](#))' with three radio button options:

- Upgrade (recommended)
This option upgrades your solution to the latest version and rolls up all previous patches in one step. Any components associated to the previous solution version that are not in the newer solution version will be deleted.
- Stage for Upgrade
This option upgrades your solution to the higher version, but defers the deletion of the previous version and any related patches until you apply a solution upgrade later.
- Update (not recommended)
This option replaces your solution with this version. Components that are not in the newer solution will not be deleted and will remain in the system.

Below this is another section titled 'Previous customizations on components included in this solution ([Learn more](#))' with two radio button options:

- Maintain customizations (recommended)
This option maintains any unmanaged customizations performed on components, but also implies that some of the updates included in this solution will not take effect.
- Overwrite customizations (not recommended)
This option overwrites or removes any unmanaged customizations previously performed on components in this solution. This option does not affect components that support merge behavior (forms, sitemap, ribbon, app modules). Components that have other managed solutions on top of the existing solution you are replacing do also still remain on top and are not affected by this option.

Figure 5.4 - Solution update options

Selecting one of the first two **Solution Actions** 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.

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.

Learning about 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 a 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.

Learning about major Power Platform updates

The platform updates are provided bi-yearly (*spring and fall 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.

Updates of first-party applications

The **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**.

Power Platform hotfixes

The **hotfixes** are unattended small updates provided by Microsoft very frequently and are installed automatically on every environment within a specified time frame. Hotfix announcements are made within the *Microsoft 365 administration* (portal and mobile apps) and as knowledge base articles.

Note

For Knowledge base articles please refer to the following link: <https://docs.microsoft.com/en-us/business-applications-release-notes/dynamics/released-versions/dynamics-365ce>

Release notes for Power Platform Hotfixes are available at <https://docs.microsoft.com/en-us/business-applications-release-notes/powerplatform/released-versions/powerapps>

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 the **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, which 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.

Getting an 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.

Understanding helper and quality check tasks

There are the following helper tasks as part of the package:

- **Power Platform Tool Installer:** This needs to be the first task in every pipeline to ensure the Power Platform Tools supporting artifacts, required by the DevOps agent, are installed properly.
- **PowerApps 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 *PowerApps Checker tool*.

Understanding 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 not any support for automatically creating solution packages. There are currently the following tasks available:

- **Power Platform Export Solution:** Task can export an existing solution from an environment as unmanaged or managed.
- **Power Platform Import Solution:** Task can import a solution package into an environment.
- **Power Platform Unpack Solution:** This can unpack a managed or unmanaged solution into the files contained in the package.
- **Power Platform Pack Solution:** This can pack solution files into an unmanaged or managed solution package.
- **Power Platform Set Solution Version:** This can set a version number of an existing solution in an environment.
- **Power Platform Deploy Package:** This can deploy a package created with the *Package Deployer Tool* into an environment.
- **Power Platform Publish Customizations:** This can publish all customizations in an environment.

The solution tasks are key to create deployment automation between Power Platform environments.

Understanding environment tasks

Environment tasks can be used for various manipulations of the Power Platform environments. There are currently the following tasks 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 a user-friendly name.
- **Power Platform Reset Environment:** This can recreate an existing environment. This task, in one step, deletes an environment and then creates a new environment with the same parameters as the deleted one.
- **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.

Using 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 describe some of them.

Committing a solution to the source control

One of the most usual 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. A typical solution commit procedure for an individual developer or for a consolidation environment could look like the following:

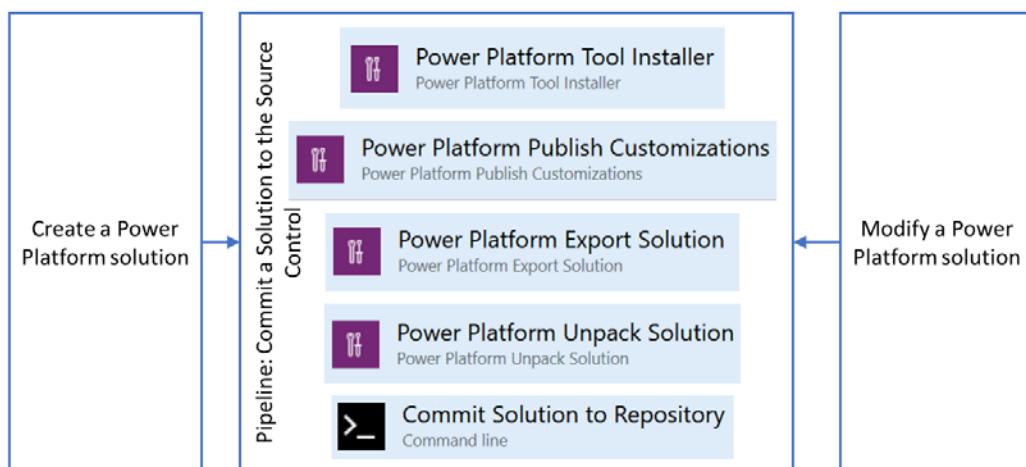


Figure 5.5 - Committing a solution to the source control

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 be always 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 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:

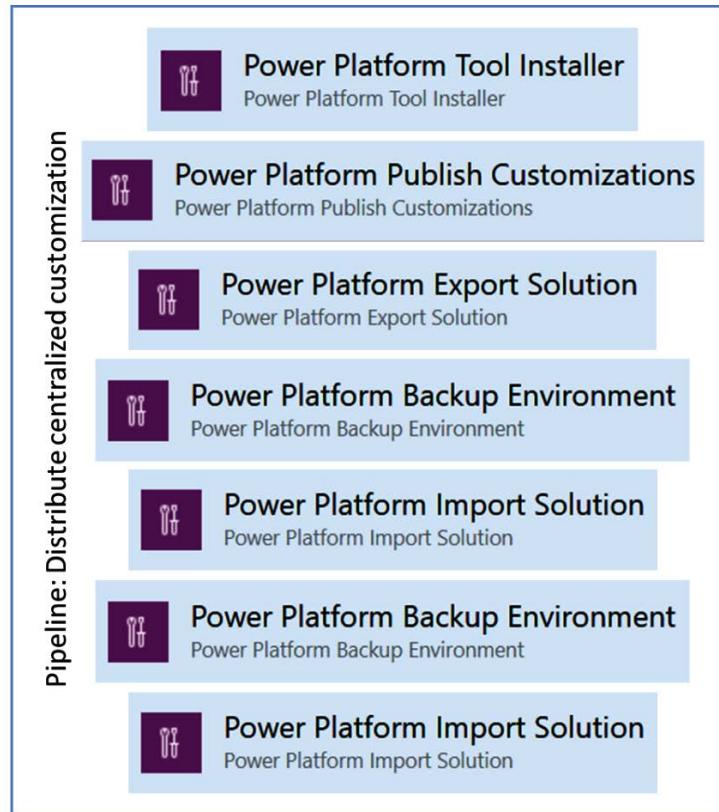


Figure 5.6 - Distribute centralized customization

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:

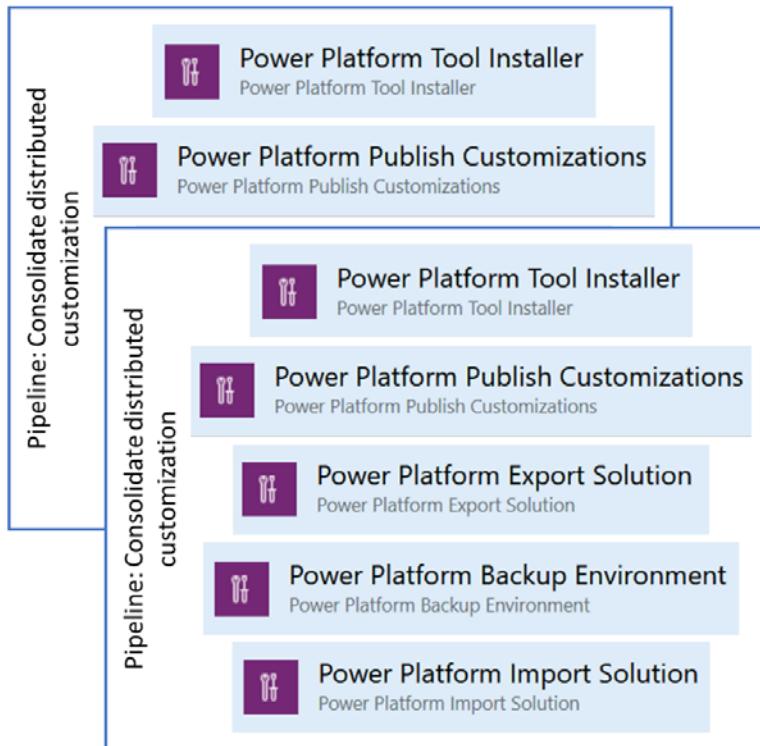


Figure 5.7 - Consolidate distributed customization

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:

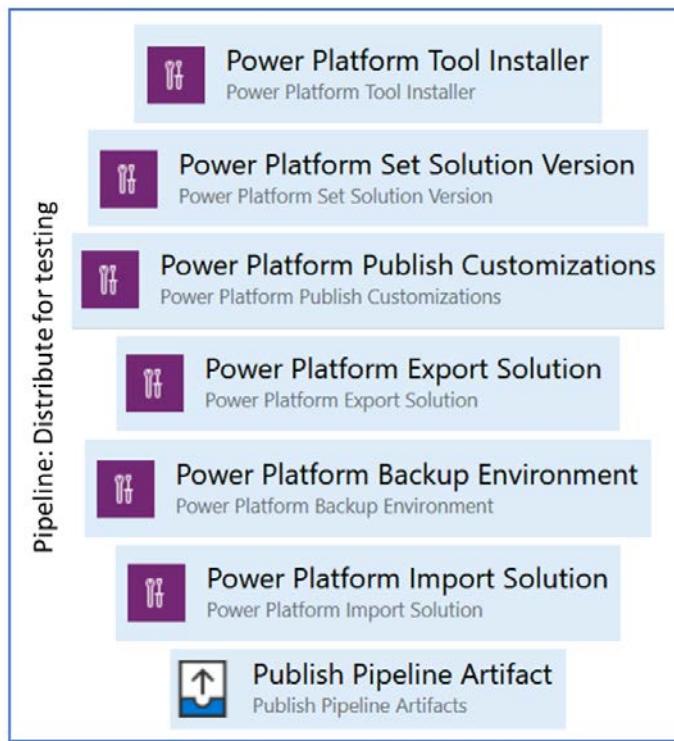


Figure 5.8 - Distribute for testing

In this scenario, the consolidated solution usually gets a new solution version number. The solution is exported as managed from the consolidation environment and imported into the testing environment. 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. It needs to be decided, based on the solution complexity, whether it is sufficient to use, for all Azure DevOps ALM automations, 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.

Application lifecycle management for Power BI

As we already learned in this chapter, Power BI components cannot be included in Power Platform solution packages. Moreover, the Power BI development tools do not provide integration with ALM tools like Azure DevOps yet. The consequence is that there is no possibility to automatically check dependencies between Power BI solution components and the other parts of a Power Platform solution.

So, generally speaking, keeping a strict ALM approach for Power BI is quite challenging. There are, however, certain possibilities to implement an ALM approach explained in the following sections.

Understanding environments in Power BI

Power BI is not related to the Power Platform environments by any means, so that a staging cascade for developing, testing, and deploying Power BI components must be implemented in a different way. It is recommended to use the concept of **Power BI workspaces** for this purpose, as illustrated in the following diagram:

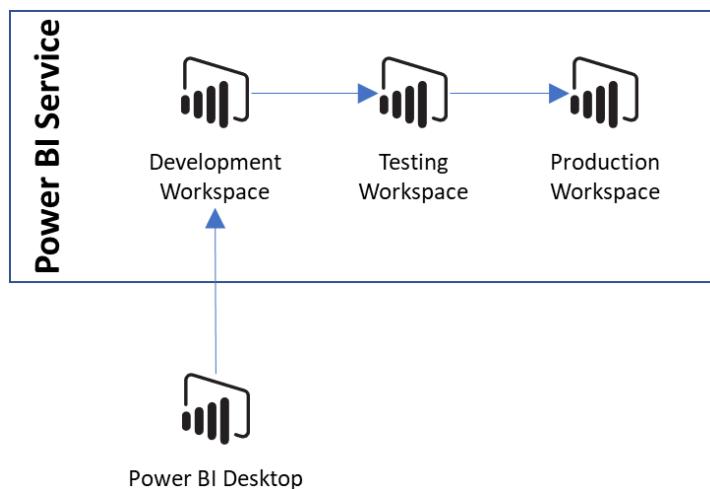


Figure 5.9 - Power BI ALM environments

For every staging environment required for the ALM, a separate workspace can be created.

Learning about Power BI components

Despite the complexity of Power BI, there are just two components types that can be used in an ALM approach:

- **Power BI file (PBIX):** It contains all the Power BI artifacts created with Power BI Desktop, including the **datasets** with data.
- **Power BI Template (PBIT):** It contains all Power BI artifacts but **without** any data.

Learning about the 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 to automatically deploy the files between development, testing, and production. Therefore, we can use an approach such as the one illustrated in the following diagram:

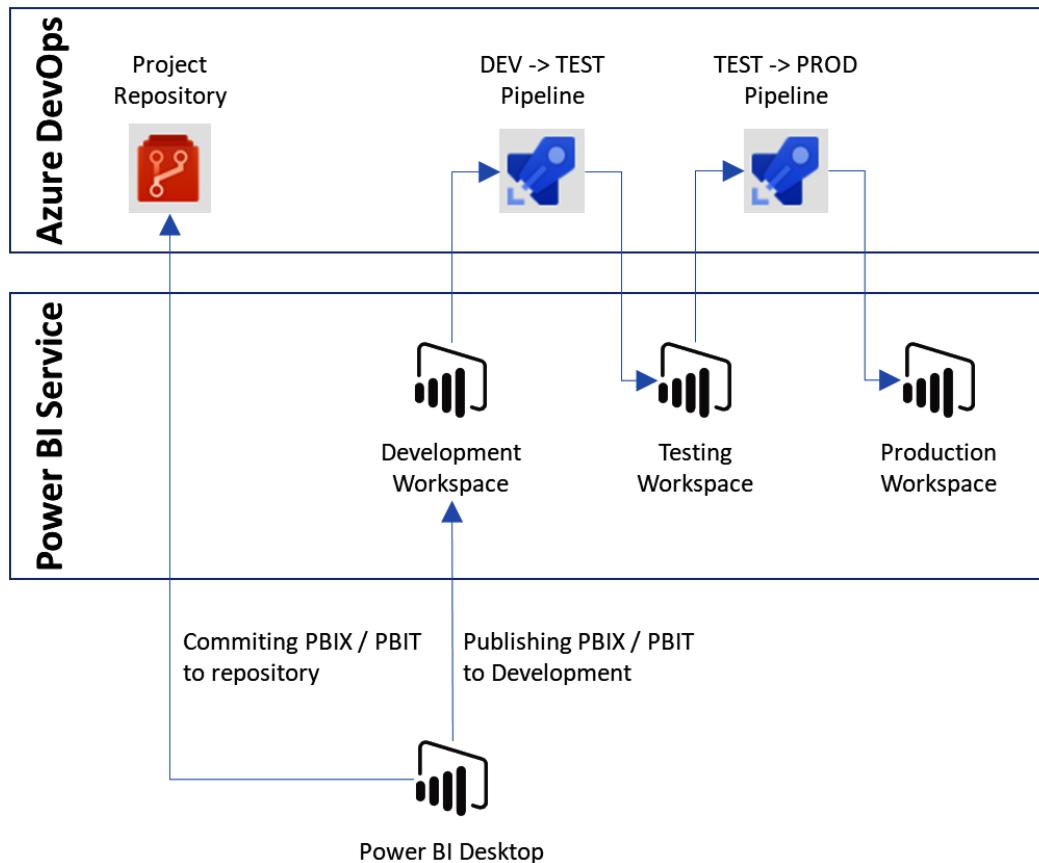


Figure 5.10 - Power BI ALM approach

As we can see in the diagram, the respective components and processes can work in the following way:

- 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.
- At the same time, the developers publish the solution components into the **Power BI Service** in the **Development Workspace** to test the functionality from developer standpoint.
- 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.
- An **Azure DevOps** deployment pipeline exports the components from the **Testing Workspace** and imports them into the **Production Workspace** for production use.
- 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.

Learning about Power BI deployment pipelines

For the Power BI Premium subscription, there is a new feature called **Power BI deployment pipelines**, which as of the writing of this book is in preview. 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 pipeline
- Move content between pipelines (for example, from a Development pipeline to a Test pipeline and finally to a Production pipeline)

Important note

This capability is in preview and does not yet enable integration with Azure DevOps. It is recommended to use the *ALM approach* using workspaces until this new capability matures.

Application lifecycle management for other solution components

Since a whole Power Platform solution can consist of many other solution components, the development needs to be supported with ALM as well. Those are specifically the following:

- Microsoft Azure solution components
- Custom development components for on-premise deployment

These types of components are **custom development**, which makes them easy to be managed by Azure DevOps. But since they cannot be included in Power Platform solution packages, 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*: the *solution management* and the use of *Azure DevOps*. In this section, you will learn several best practices on 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 on how to properly use solution types, structure and segment solutions, and publishers.

General practices

It is best practice to always create a specific solution package for a defined purpose and to 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 but never custom components. Custom components should always be 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 recommendation from Microsoft regarding the use of the proper solution type:

- **Unmanaged solutions:** It should be used **only** within development environments, regardless of how many they are and how complex they are structured.
- **Managed solutions:** It 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 proper use of the 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 solutions.

Using a single solution

Using a **single solution** package for the whole Power Platform solution is a viable way for most 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 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.

Using 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:

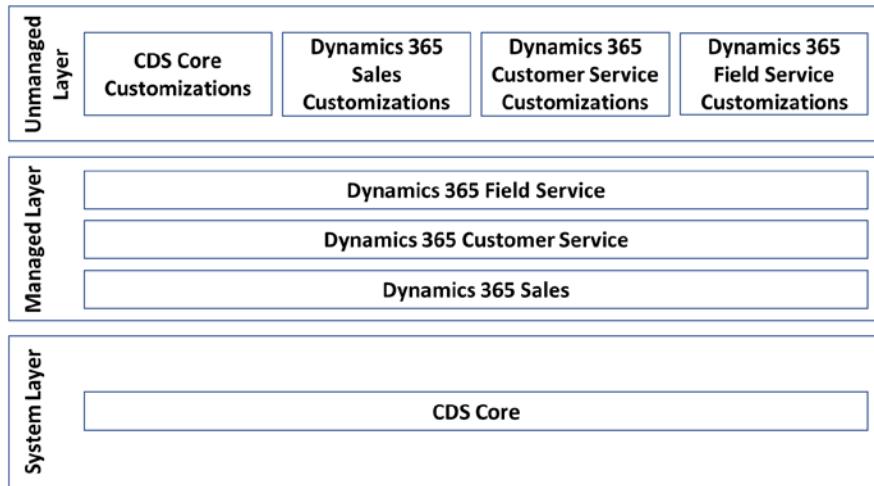


Figure 5.11 - Workload-based multiple solutions approach

Another example of implementing multiple solutions could be for a **global solution**, deployed in 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:

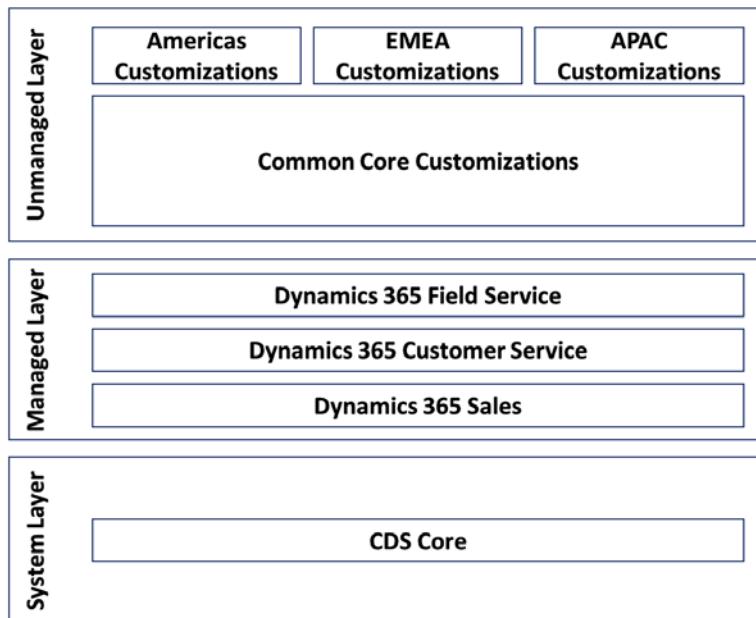


Figure 5.12 - Regional-based multiple solutions approach

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 case 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(s). These solution packages would usually have their own independent ALM and the whole management of them 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

When using **segmentation**, there are three options defining what can be included from an entity in a solution, as illustrated in the following screenshot:

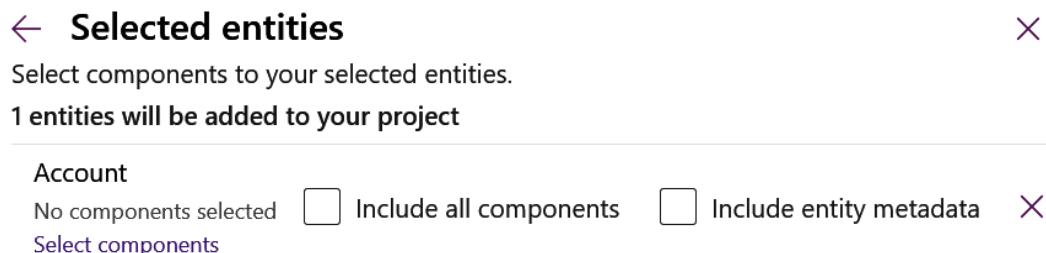


Figure 5.13 - Segmentation options

It is important to understand what these three options do exactly and what are the best practices for using them:

- **Include all components:** It will add all components (metadata, fields, relationships, forms, view, charts, and others) of an entity into the solution package. This option is used by default when creating new entities within a solution package. For existing entities, this option should only be used if that entity is not yet present in the target environment.
- **Include entity metadata:** It will add only the entity with its metadata, which is a collection of basic and behavioral settings for the entity, but no entity components. This option should only be selected when there is a need to change some of the entity settings in the target environment.
- **Select components:** It is the true solution segmentation capability because clicking on this link opens a dialog where any entity 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 an entity 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 directly 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 publisher 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:

- The separation of datasets and visualizations, where the datasets can be created by data experts knowing 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 the visual consistency of the created content
- Using preferably 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 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 Microsoft AppSource.
- There will be no region-specific deviations of the central Power Platform solution.

The preliminary solution structure was specified according to the following diagram:

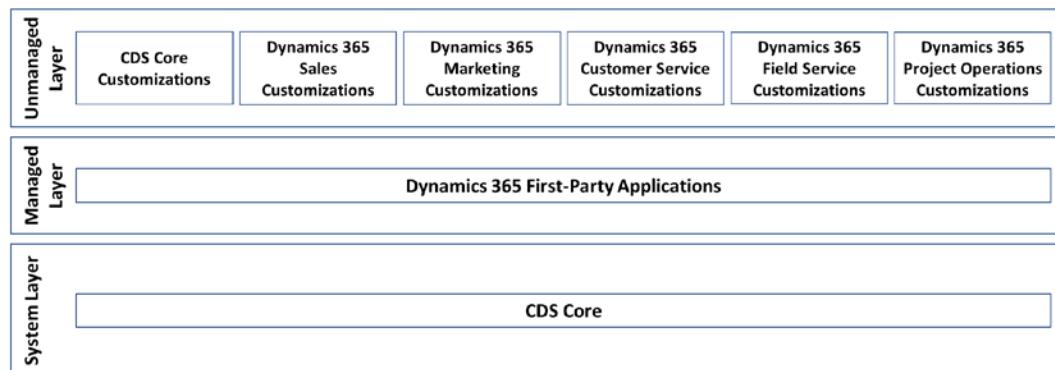


Figure 5.14 - Contoso Inc. Power Platform solutions structure

Contoso Inc. decided that the project team needs to ensure to 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 own 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.

Section 3: 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 a 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 lifecycle.

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

- Getting an overview of the implementation approach
- Understanding customer enterprise architecture and environment
- 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 three 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 lifecycle 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 of running internal IT projects, with Power Platform, this is a new experience, and they would like to prepare everything for a successful implementation project.

Getting an overview of the implementation approach

There are many different IT implementation project types, from pure infrastructure projects to implementing robotics, artificial intelligence (AI), 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 is in the following areas:

- **Modernization:** Replace a 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 automation, AI, mixed reality, and more.
- **Mobility:** Empower mobile workers with the same capabilities as traditional desktop workers.
- **Cost savings:** Change the traditional *CapEx* model to *OpEx*, saving significant upfront costs of a business solution.

- **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 on IT.

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.

Following 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 the 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 be deeply integrated into the existing ecosystem and it needs to play well inside the same. 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 the Microsoft technology in general?
- Is the organization prepared to accept a cloud solution?
- Does the organization require the cloud solution to reside in a certain geographical region or country?

- What kind of authentication ecosystem is established? Is it Microsoft 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 existing 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 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 more suitable for running batch jobs?
- What is the level of maturity of the organization's adoption and change management? Are they able to ensure 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, what are the areas with possibly increased efforts, and what are the major project risks and how to mitigate them.

Let's take a few examples of typical customer requirements in the next sections.

Data residency example

The organization requires the cloud solution to be located in **Germany**. This is currently not possible since the Microsoft cloud region **Germany** is not yet Power Platform-enabled.

The solution would be to temporarily select some other Power Platform-enabled region within the EU, for example, **Europe** or **France**. As soon as **Germany** is enabled for Power Platform, a move from the temporary region into the **Germany** 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.

Authentication provider example

The organization does not rely on Microsoft Active Directory as their main authentication provider, preferring 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 impose additional efforts to implement advanced security within Azure Active Directory alone and to enable a cloud-only single-sign on capability.

Internet access example

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 is routed through the mobile *VPN solution*.

Data protection example

The organization does not trust the **Microsoft encryption technology** used in the *Power Platform databases (CDS database)* and requires the use of its **own encryption keys**.

This will require first becoming eligible for a **Bring-Your-Own-Key (BYOK)** solution and then planning for additional efforts 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, and tools used for Power Platform implementation projects as well as about 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 clearly focus on the typical and most frequently used Power Platform-related methodologies.

Understanding programs and projects

It is important to understand the terms **program** and **project** and be able to distinguish between them in the context of Power Platform implementation:

- A **project** is a temporary undertaking to create a 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 separate 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 better known and IT-relevant methodologies and approaches are as follows:

- **Waterfall:** This is the traditional non-agile methodology.
- **Project Management Institute PMBOK:** This was adopted by Microsoft for their own *SureStep 365* methodology dedicated to Dynamics 365 implementations.
- **Agile:** Agile, with its modifications and extensions such as Scrum, Kanban, Scrumban, and Rapid Application Development, is used for smaller and fast-paced projects with very self-organized project teams

- **Extreme Programming:** This is used rather for smaller and very agile projects.
- **Adaptive Project Framework:** This is used for projects with a lot of unknowns to enable adjustments in scope.
- **PRINCE2:** This is mostly popular in connection with 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, popular in the implementation of both hardware and software solutions within a single project.
- **Iterative model:** This is a less dynamic methodology, where certain phases are waterfall and others agile, and where there is a typical overlap of iterations.

Every software implementation project, regardless of the methodology or approach used, has certain major phases, which might contain sub-phases. Following are the typical phases of a software implementation project:

- **Requirements gathering**, where the customer requirements are collected and documented
- **Analysis**, where the collected requirements are analyzed and transformed into data models, business processes, and other artifacts
- **Design**, where the results of the analysis phase are transformed into the technical architecture and design of the solution
- **Development**, where the technical design is implemented into working solution components
- **Testing**, where the solution components are tested and validated against the requirements
- **Deployment**, where 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 main project steps and activities happen sequentially, as illustrated in the following diagram:

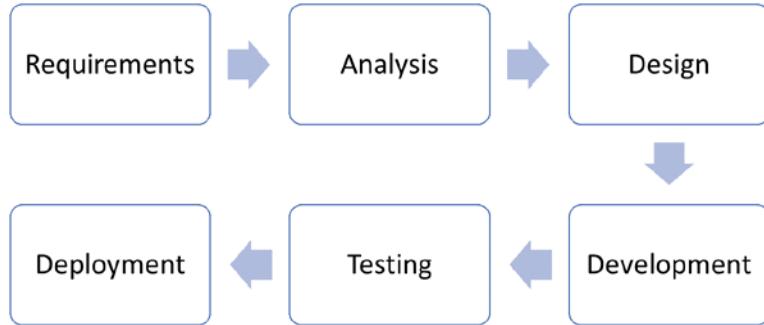


Figure 6.1 - The Waterfall project approach

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 once full development is finished, which imposes the risk of scope creep.

For Power Platform, this project model is best suited for deep technological and, at the same time, relatively independent parts of the program or project:

- **Infrastructure project or sub-project:** This is specifically for on-premises deployments or for deployments with on-premises components.
- **Integration project or sub-project:** This is specifically for very complex integrations with various on-premises or third-party systems, with significant dependencies and implementation efforts on the legacy side.
- **Data migration project or sub-project:** This is specifically 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 quickly 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:

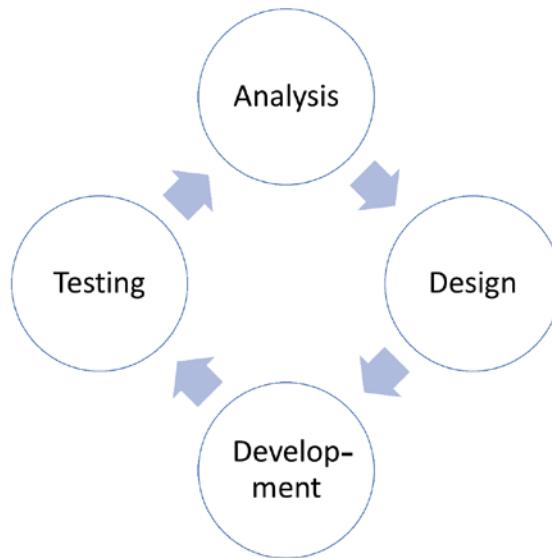


Figure 6.2 - The agile project approach

The agile model has the following main features:

- The model uses just light documentation.
- The sprints are short periods of time, usually 2 weeks.
- Agile teams are very self-organized. They decide the content of each sprint based on the overall situation and resource availability rather than 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 for smaller projects with limited software development, integration, and data migration requirements and with little 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:

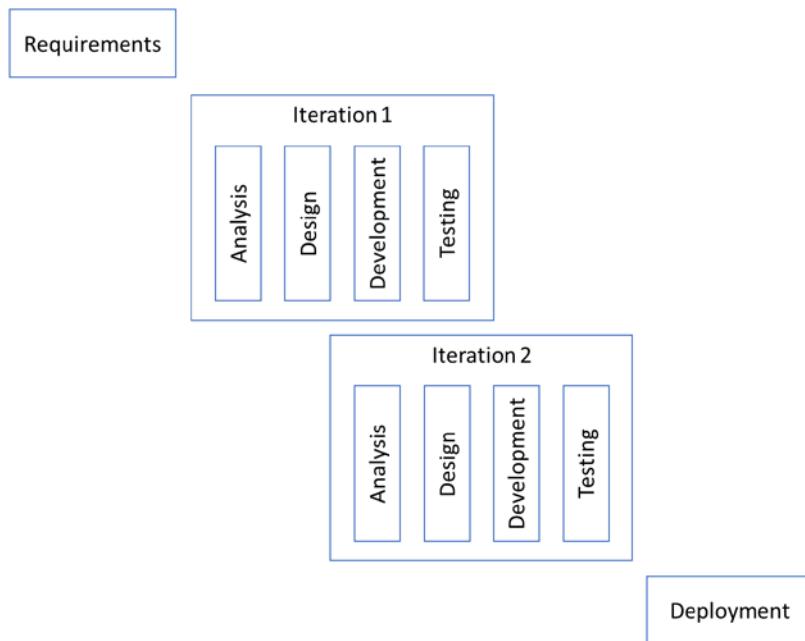


Figure 6.3 - The Iterative project approach

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 4 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-premises, 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 the 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 and tools 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 earlier in the preparation of a project. It is a rather high-level, interval-type of estimate (for example, estimating the effort as an interval of 75%–125%) 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

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 respective 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 requirements estimation example

In the preceding example, a requirement of the complexity level **Simple** is estimated by default as **40 hours** total effort and a requirement of the complexity level **Complex** as **120 hours** total effort. The split into 4 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 the requirements against 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. It is important to consider that some infrastructure might be necessary already 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 or software/cloud licenses might need to be purchased from hardware 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 integration**, 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
- **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 existing on-premises or cloud IT system 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 the 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 an 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 entities should the data be migrated?
- What is the overall volume of the data?
- What is the available time window to execute the data migration from business point of view?

Additionally, the best way to estimate the data migration is to split down the estimation to the individual entity level. Consider the preceding list and assess every entity 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 into the total estimate in the correct project phase. Following 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 deployment of the accepted solution packages to production environments and final data migration
- **Localization**, which needs to be part of the solution development for multilingual solutions
- **Trainings**, which are 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 the mentioned efforts are usually estimated individually based on solution complexity, customer requirements, collective experience, and other factors.

Learning about 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 the 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, although there is no native integration with the Project desktop. Since this product is based on the common data service, 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. Aside from the application lifecycle management capabilities discussed in the previous chapter, the product also offers project management capabilities. It is possible to leverage the following capabilities:

- **Perform a general project configuration action** such as select the project process type (Basic, Agile, Scrum, or CMMI), configure project teams and permissions, configure sprints and assign them to teams, and configure 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 to create certain project documentation. In this section, we will describe the most typical project-related documentation used in Power Platform implementation projects.

Creating a 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 their 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:

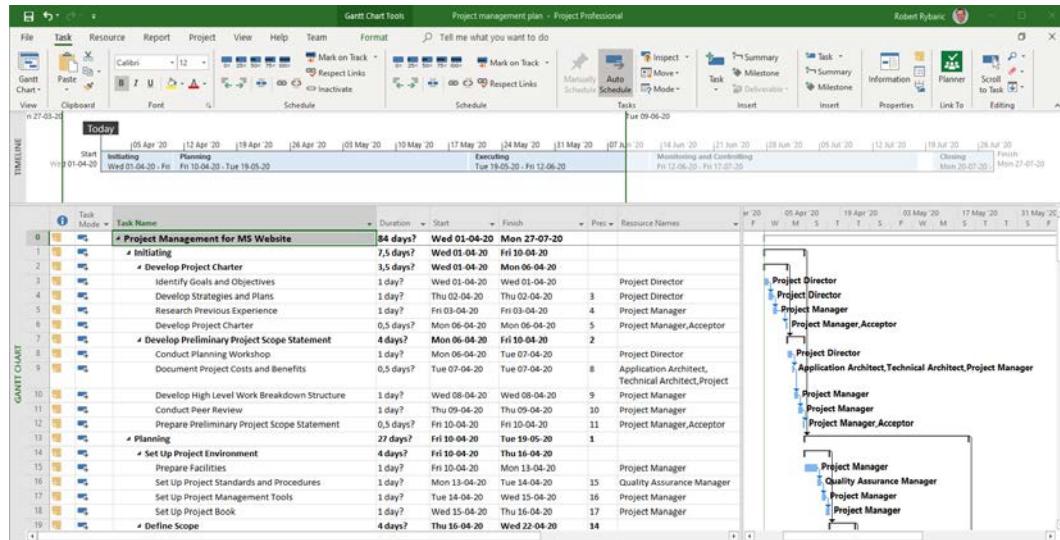


Figure 6.5 - A project plan example

In the preceding example, there are certain project phases with subphase(s) and project activities. For every project activity, there is a start and end date and a duration along with the resource assignment. On the right side, there is a Gantt chart graphically representing the project flow.

The requirements document

A 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 requirements document in its initial state is often used as a source for the project effort estimation. When finalized, the document is used as a basis 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 with data streams, and others)
- Any other required architecture domain documentation

A more detailed technical document, created later during project execution, is the solution or technical design document, and is described in the next section.

The solution/technical design document

The solution or technical design document goes deeper into specifying the solution in the following areas:

- Data model design (a full data model of the CDS database covering attributes, relationships, and any other database components used in the solution)
- User interface design (a design of model-driven apps, canvas apps, PowerApps 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 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 details, specifically the inputs, test conditions, testing procedure, and expected results.

Another usual documentation type is documentation for training the users and administrators as regards the solution, as described in the following sections.

Training documentation

For the purpose of formalized end users or key-user trainings, training documentation might be required that contains the training approach and training materials. 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 the 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.

Knowing 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 assign 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 with running software implementation projects
- Availability of internal resources that can be assigned full time or dedicated part 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

In case 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 upfront.
- Fewer 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 there is a lot of deep expertise and experience needed to be able to successfully implement such a solution.

Understanding project roles and responsibilities

The project 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 very complex projects might need a *multi-step hierarchy* with a large number of resources on both the customer and the implementation partner sides. An example of a project setup is illustrated in the following project organization structure diagram:

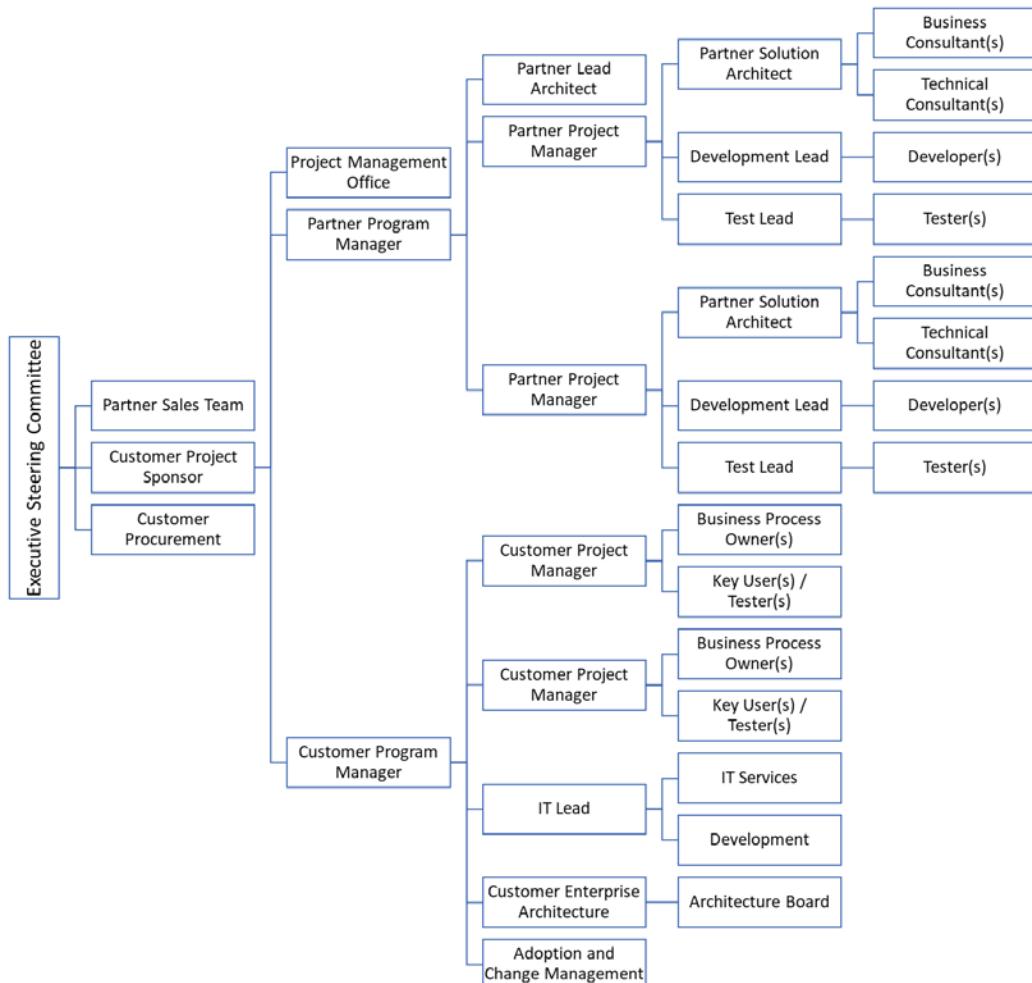


Figure 6.6 - Complex Power Platform project organization structure

Figure 6.6 illustrates a project organization structure with six levels of hierarchy. This complexity is not unusual for large Power Platform implementations.

In this section, we will cover the roles and responsibilities of a typical large project in more detail

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. 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 have influence on the whole organization.

Customer project sponsor

The **customer project sponsor** is usually a representative from higher management who serves at 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 be involved to a certain degree during execution as well to ensure that 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 the 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 controlling

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 a situation 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 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 for 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 a situation where the solution implementation is subdivided into multiple projects. In such a 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 established.

Solution architect

Solution architect is a key role for every project, coordinating the solution development from the customer requirements and technology point of view, with the following responsibilities:

- Coordinating the translation of 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, CDS workflows, business rules, plugins, and Azure Functions
- 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 roles 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 trainings, and other such activities. The responsibilities of a business consultant within a Power Platform implementation project may include the following:

- 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 CDS, 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 a situation 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 resources. The responsibilities of a technical consultant within a Power Platform implementation project may include the following:

- Creating technical design documents
- Performing customization of CDS, 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, there is a need to include developers in the team. A development team in a project usually consists of a **development lead** and **developers**. A 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 larger development team, the individual developers might work on individual tasks or even on larger 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. 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 customers
- 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, there can be any number of other specialized roles necessary 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).
- **Release manager:** This role is responsible for application lifecycle management activities (setting up Azure DevOps components such as the repository, pipelines, performing solution distribution and other similar activities). This role can also be assigned to the development lead or test lead.
- **Integration lead:** This role is responsible specifically for the development of the integration solution (responsibility separate from the development lead and test lead).
- The **data migration lead** is responsible specifically for the development of the data migration solution (responsibility split from the development lead and test lead).

Customer roles and responsibilities

One of the key factors as to why software implementation projects fail is a lack of customer commitment. Large companies with experience in 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 either a temporary assignment or a permanent role, for example, as part of the customer's PMO. In such a case, the program manager permanently 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, as for 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 assume some responsibility in 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 respective IT areas to the project
- Providing support in developing the solution's technical architecture

IT services

During a Power Platform implementation, there is a need for various support services from the customer's IT organization. There are the following roles that 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 provided by 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 include 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 in relation to their 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 a customer's IT systems
- Developing security solutions to integrate Power Platform security into a customer's IT corporate security

Enterprise architecture

Many large organizations have an **Enterprise Architecture (EA)** department or team, 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 project preparation by giving guidance on the best solution approaches from an EA point of view. During project delivery, the EA team ensures that 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 factors 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 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 lifecycle with all possible phases, from the initial concept until a solution is brought into daily operation.

Understanding project phases

On a high level, a project consists of three main phases and each phase can consist of many subphases. Not all of these subphases are always present or in the same order as that described in the following sections, but the following structure represents a typical large Power Platform implementation project lifecycle.

In this section, we will provide a detailed description of the typical activities and outputs in all of the subphases of such a project for both the customer and the implementation partner (consulting company).

Knowing 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, there is an idea or **demand** for a new IT solution identified in a company. The reasons for an identified demand can be very different. It can be purely an idea that some business processes need to be automated, as it is no more 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 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 the whole subsequent steps and the whole project preparation and execution process.

After the demand is identified, an internal discussion and evaluation usually start and lead to a potential decision that the demand is legitimate. A usual next step can be 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.

In the next part of this section, we will describe the content of these main feasibility study areas.

Economic feasibility

The **economic feasibility** evaluates the potential business and financial benefits, cost/benefit analysis, expected **Return Of Investment (ROI)**, financial projections, risks analysis, and finally a high-level cost estimate.

Technical feasibility

Technical feasibility evaluates the respective 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 comprising resource availability, business closures, legal and financial reporting periods, and more. 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
- Internal resource allocation
- High-level project schedule
- High-level technology decision
- The vendor and implementation partner selection process

Specifying budget

In case of a go decision and a high-level budget decision, there could be a further budget specification, such as budget framing, the possibility to extend 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 in the preparation phase is final approval by top management to start the procurement process for 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 possible and optional step could be issuing an **RFI**. The purpose of the RFI is to gather comparable information from a broader selection of vendors or implementation partners about possible products and services they can offer to fulfill requirements. Compared with the next step, the RFI requires fewer details from 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 in a strict formalized structure with clear and simple response possibilities. 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
- Basic schedule and pricing estimates

The evaluation of the RFI responses leads usually to a shortlisted number of pre-selected vendors or implementation partners who 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, or RFT are names for similar types of the 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
- A high-level solution architecture
- A 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 from the various experts required to support answering the request.

The customer sometimes requires the proposal or quotation to be presented 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 can be a request to 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 certain 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 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 certain 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

Following are some additional details to 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 for 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 already described earlier in this chapter.
- The **project schedule** is created with respect to the effort estimate to provide an understanding about the project duration, phases, and so on.
- The customer may require various additional response materials, 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 a final vendor or implementation partner selection 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 discovery is to better understand the customer's requirements and the vision of the required solution. Discovery is usually a short activity requiring days or several 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 project execution since, after the contract signing, the customer is usually keen to embark on execution of the project as soon as possible.

Contract

The **contract** signing is the final project preparation step. Immediately after the signing, project execution starts.

Project execution phase

The **project execution** phase is the contractual phase of the project, where the implementation partner, in cooperation with the customer, performs all the work to fulfill the project's 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 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 already 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.
- 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 material 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 **project initiation**. The main activity within this phase is, besides numerous 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 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 agile than the true agile approach, there is a non-iterative **initial analysis** phase used to prepare the project for the iterative execution phase. The initial analysis usually takes several weeks and consists of the following typical tasks:

- A series of business-related workshops dedicated to workload categories is conducted 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.
- A series of technical workshops dedicated to technical architecture, infrastructure, security, integration, data migration and other technical topics is conducted. The typical participants are project managers, the partner solution architect and technical consultant(s), the customer IT lead, enterprise architecture, and other relevant roles.
- The out-of-the-box capabilities of the Power Platform and Dynamics 365 components, along with documenting gaps against requirements, are presented.
- The requirements document is refined and all requirements validated with the customer.
- The requirements structure is included in the Azure DevOps environment.

- The content is finalized for each of the planned iterations for the iterative execution phase.
- The project plan is updated.
- Initial versions of the solution architecture, solution design, and other main project documents are created.

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

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 subphases: analysis, design, development, and testing.

The subphases 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 paragraphs.

Iterative analysis

The main purpose of the iterative analysis subphase 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 the implementation approach for the requirements.
- Proceed in the technical preparation for the subsequent subphases (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 subphase, there is clear and documented scope for the subsequent iterative design subphase.

Iterative design

The main purpose of the **iterative design** subphase is to define how to implement all of the requirements analyzed and documented in the iterative analysis subphase. 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 CDS data model extensions
- Specifications for creating model-driven applications
- Specifications for 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 **PowerApps Component Framework (PCF)** controls
- Specifications for developing custom Power Automate connectors
- Specifications for creating automations with PlugIns and custom workflow actions
- Specifications for creating canvas apps

- Specifications for creating a 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 subphase, there is clear and detailed documentation for the subsequent iterative development subphase.

Iterative development

The main purpose of the **iterative development** subphase 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 CDS database, model-driven apps, canvas apps, Power Automate, Power BI, and so on.
- Perform custom development tasks in every part of the overall solution (CDS, PCF components, custom connectors, Azure, and on-premises).
- Develop data migration scripts.
- Develop test scripts.
- Perform unit testing of the developed artifacts.
- Perform application lifecycle management activities (solution export/import/commit into repository and custom development commit).
- Update the project plan, Azure DevOps board, solution architecture, solution design, and other main project documents.

At the end of the iterative development subphase, there is a set of solution configurations, customizations, and custom development artifacts prepared for the subsequent iterative testing subphase.

Iterative testing

The main purpose of the **iterative testing** subphase is to perform all necessary testing of the artifacts developed in the iterative development sub-phase. While the iterative development subphase contains unit testing as part of the development process, iterative testing consists of a series of higher-level testing activities. Moreover, the difference is that the iterative testing activities are performed on a testing 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 materials.
- 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 subphase, 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 deployment. This phase usually takes several weeks, and the main activity is the official UAT. The testing activities are performed on 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 the data.
- Perform a key user/train-the-trainer training.
- Perform the UAT.
- Perform the final data migration tests.
- Finalize the 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 project phase, everything is prepared for the final 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 on the production environment.
- The final data 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 that the customer's operational team is 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 such cases, 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.

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 production use of the Power Platform solution 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-premises components and, as such, does not need extensive maintenance. Following are typical maintenance activities that must be covered by the customer's support organization:

- **Maintaining Microsoft product updates**, especially installing updates for first-party applications (Microsoft Dynamics 365) that are not installed automatically
- **Manual backups** of the Power Platform environments, in case 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** 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 second-level support is usually support of 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 a certain point of time. In case a Power Platform solution should be decommissioned and replaced with any other solution, there is usually one single concern, regarding the data managed in the solution. A Power Platform solution based on the CDS 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 CDS 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 to select the best suitable implementation partner and start the project. They have made a series of preparation activities and have made some important decisions.

Bidding process

Since Contoso Inc. already decided upfront to implement a Power Platform solution, they triggered a bidding process for an implementation partner only. After a structured process consisting of an RFI, RFP, a proof of concept and a 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 Microsoft partner level of Gold Certified Partner for the Microsoft Dynamics 365 and Power Platform, Microsoft 365, and Microsoft Azure specializations. 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:

- **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 involve 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 the in-depth involvement of Contoso's key users in solution development.

- **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 most suitable, since the implementation of the integration solution will need longer preparation times and organization and management of many external stakeholders.
- **Power Platform data migration project** managed using the *waterfall* project implementation model. This project will cover the data 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. also 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., having program managers, project managers, and all other roles as illustrated in the following project organization diagram:

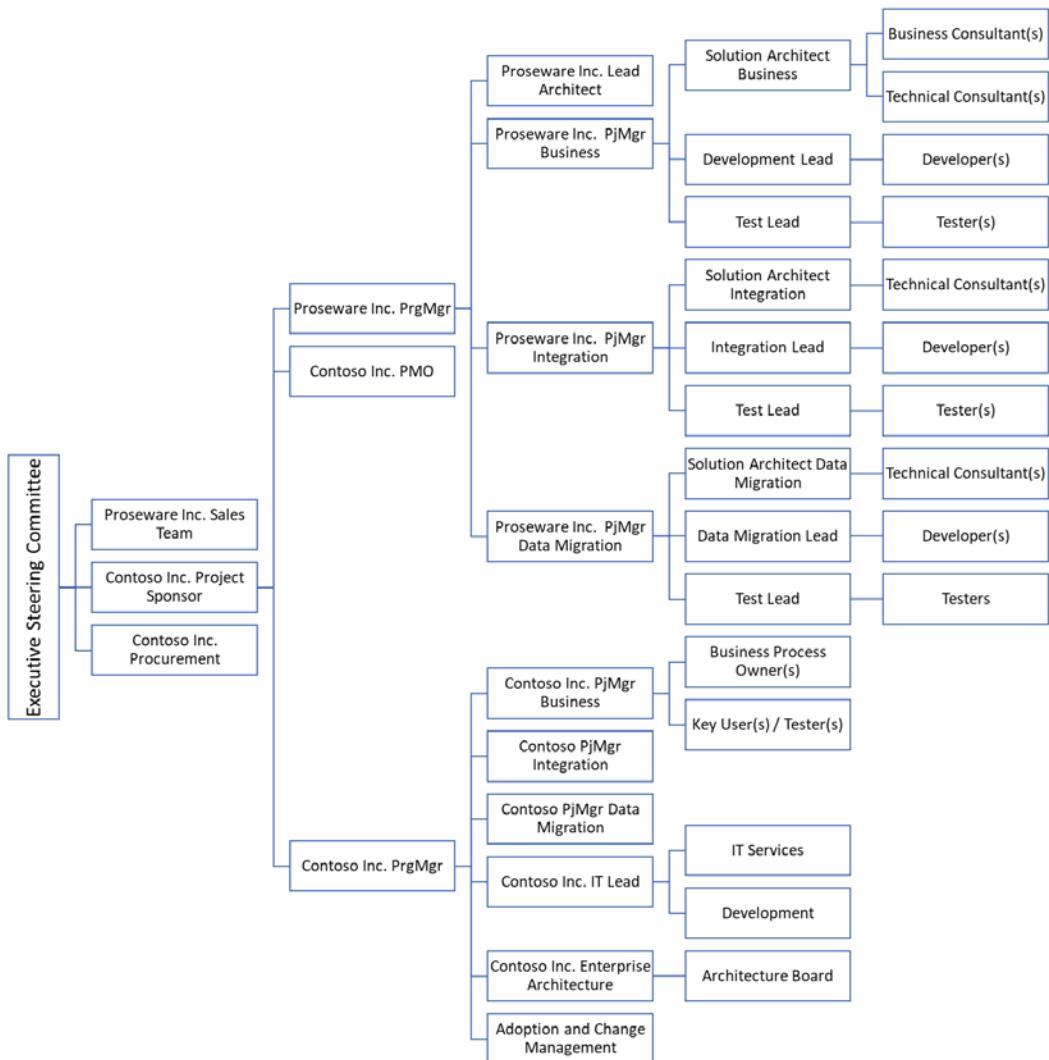


Figure 6.7 - Contoso Inc. Power Platform project organization

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 Managers (PjMgrs)**, working closely together in the assigned areas of developing the business solution, developing integration, and preparing 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 solutions will rest with 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 within their respective 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 prerequisites for a 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 on 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 lifecycle usually looks. With this knowledge, you should be able to set up a Power Platform project properly and make sure the implemented solution fulfills 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 and, 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 to protect the 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 securing 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 this can be very different, depending on the service type.

Next, we will have a look at the fundamentals of authentication and authorization in the Microsoft cloud ecosystem.

Understanding 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. The 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:

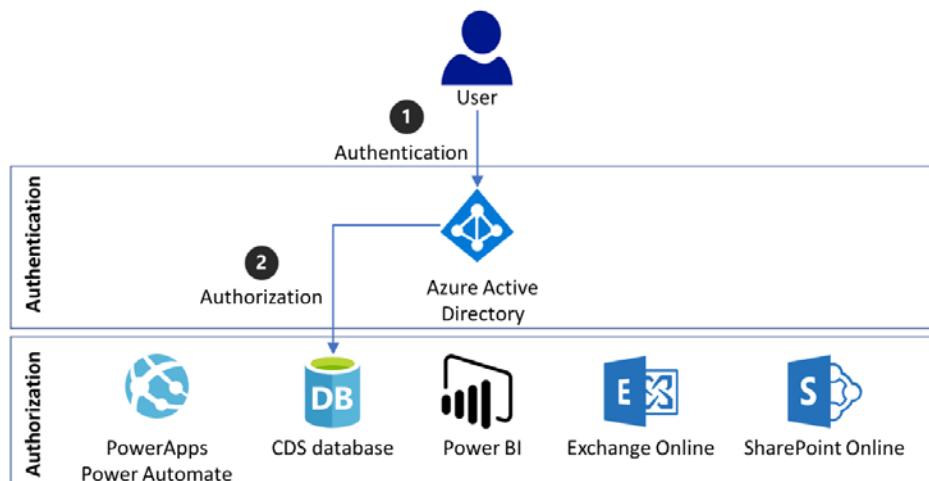


Figure 7.1 - Microsoft cloud authentication and authorization

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 the environment is managed by standard environment roles that can be assigned to users but cannot be modified.
- **CDS database applications:** Authorization within CDS applications is managed by a very detailed authorization concept consisting of business units, security roles, hierarchy security, teams, and field-level security.
- **Power BI:** Authorization within Power BI is managed using *granting access* to parts of Power BI, like the workspaces for individual users or for AAD groups. In addition, Power BI provides the possibility 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 to 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 sections, you will learn more about the options for the three basic steps for enabling a new Microsoft cloud user, as mentioned earlier in this section.

Provisioning user identity

Every Microsoft cloud (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*.

A good example of a background process that's triggered after a license assignment is a user provisioning process for Power Platform environments with the CDS database. In a CDS-based environment, the user must be created as a record in the CDS entity `systemuser`. That's why, after assigning a user a Dynamics 365 or a Power Apps license, an asynchronous background process creates a user record in that entity.

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 to the medium complexity implemented in Power BI, to a very sophisticated and detailed authorization concept in CDS 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.

Understanding 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 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.

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:

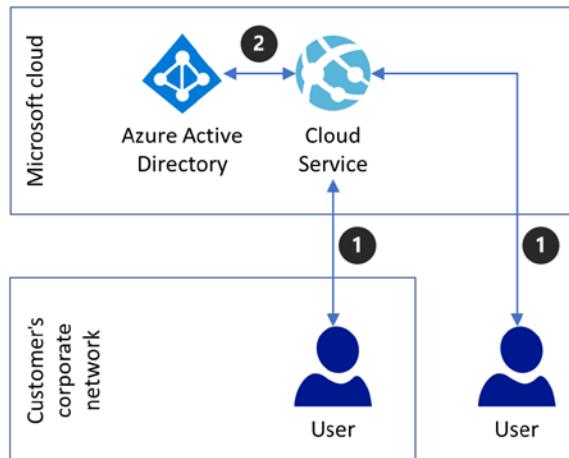


Figure 7.2 - Authentication with cloud identity

As we can see, authentication happens only in the cloud, and it 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:

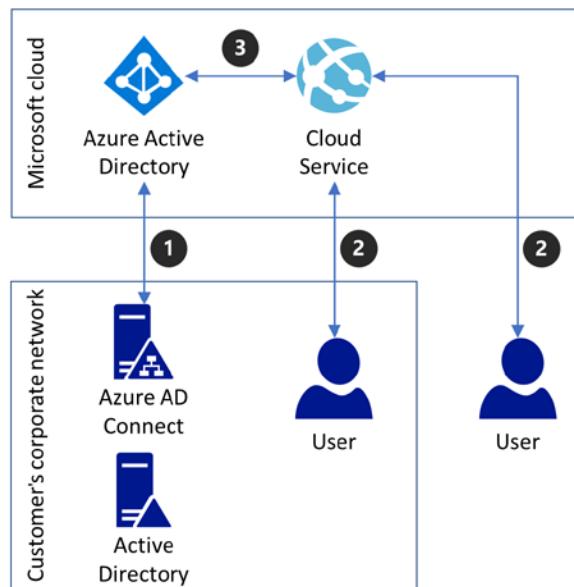


Figure 7.3 - Authentication with password hash synchronization

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 on-premises active directory. This approach is illustrated in the following diagram:

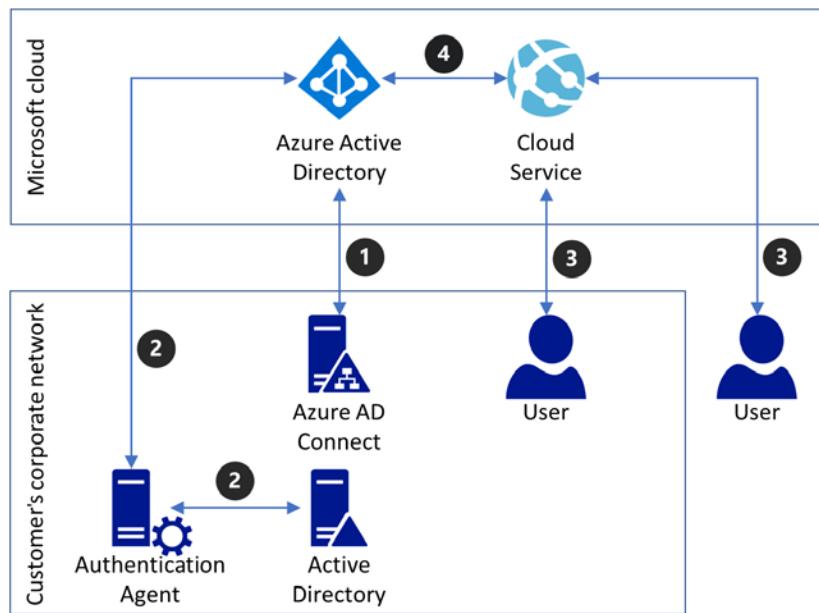


Figure 7.4 - Authentication with pass-through password verification

As we can see in the diagram, 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** 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:

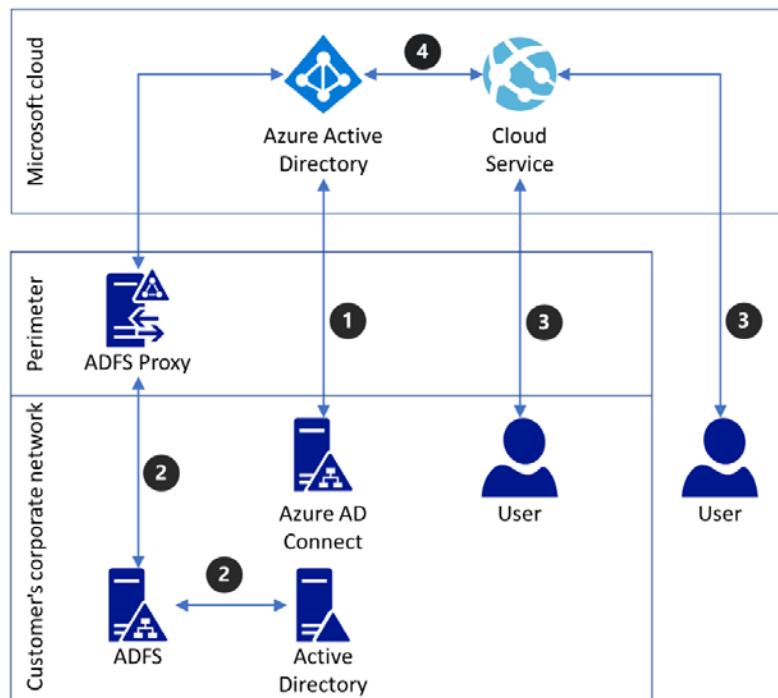


Figure 7.5 - Authentication with Active Directory Federation Services (ADFS)

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 **Active Directory Federation Service (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 of the authentication process

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 the legitimate security requirements.

- **Federation:** This approach is used by large, multinational, traditional 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.

Getting to know authentication features for internal users

AAD and ADFS are offering certain additional authentication features. These features are especially useful for Power Platform solutions. Let's learn more details about these features in the following sections.

Conditional access

Conditional access is one of the most powerful AAD security features 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 made by the conditional access component can be either to grant access, block access, or to 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*, in case 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. For details see at <https://docs.microsoft.com/en-us/azure/active-directory/authentication/concept-mfa-licensing>

Single sign-on options

For password hash synchronization and for pass-through authentication, there is an option 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, Seamless Single Sign-On is not available, since here the authentication is performed in the on-premises active directory. ADFS, however, supports single sign-on using cookies or application-specific, settings.

Important note

For federation scenarios integrating AAD with multiple independent on-premises active directory forests, there is always a selection dialogue required for the user to select the identity provider they wish to authenticate with.

Next, we will discuss a specific 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 **Software as a Service (SaaS)** solutions, which are operated on shared infrastructure and often using 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, 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 access to own tenants for Power Apps and Power Automate.

Important note

In the Power Platform ecosystem, there are certain cloud services using unique, customer-specific URLs, such as CDS/Dynamics 365 applications and other services using public and shared URLs. Using cross-tenant restrictions is useful only for services with shared URLs.

In the next section, we will focus on authentication options for service accounts, which is the opposite to user accounts.

Learning about service authentication for internal users

Authentication is equally important when building any external solution components 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.

Common Data Service authentication

Every external application that should connect to a CDS-based solution needs to be authenticated. The authentication type depends on the CDS API endpoint the application is using. There are following CDS API endpoint types: **SOAP** and **Web API (OData 4.0)**.

For the older SOAP-based endpoint, there could be authentication types including Office 365 authentication and OAuth authentication. When using the modern Web API-based endpoint, the only authentication type available is OAuth.

Important note

As of writing this book, there is a third API endpoint provided as a preview feature, the **Tabular Data Stream (TDS)** endpoint enabling read-only access to CDS data using AAD username/password authentication via port 5558.

In the next few sections, you will learn more about CDS 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, for authentication purposes, there is only the URL of the Power Apps CDS environment, a username, and a password, all of which are necessary 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. 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 standard user is illustrated in the following code snippet:

```
...
    string url = "https://contoso.crm.dynamics.com";
    string clientId = "51f81489-12ee-4a9e-aaaе-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 CDS 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 considered less secure.

The most secure authentication is OAuth with the use of application, not the standard user. 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, are 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 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 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 CDS 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 as for **Common Data Service (CDS)** since the Power BI API is REST-based and the authentication type is *OAuth*. The first step in enabling OAuth authentication is identical to CDS – register your external application with AAD. This step can be performed either directly in the Azure administration portal or using a *Power BI-specific App registration* tool provided at the following URL: <https://dev.powerbi.com/Apps>.

Similar to CDS, 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 CDS authentication would be the resource URL of the Power BI service: <https://analysis.windows.net/powerbi/api>.

Another important security option specifically for CDS-based solutions is authentication governance, which we will look at in the next section.

Studying authentication governance for internal users

Power Platform provides certain additional authentication governance features, which will be described in more detail in the following sections.

CDS 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 any specific setting, creating user accounts and assigning them Power Apps or Dynamics 365 licenses triggers the process of provisioning user account records in the `systemuser` entity into all existing CDS environments in the tenant. To avoid this unwanted behavior, it is possible to use Office 365 Groups to manage the provisioning process and select which users will be provisioned into which CDS environments. To enable this feature, following these steps are necessary:

1. Create an Office 365 group for every planned new Power Platform environment with CDS.
2. While provisioning a new Power Platform environment with CDS, assign the security 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` entity in that CDS environment.

When a user is removed from the Office 365 group, their user account is automatically deactivated in the respective CDS 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 CDS, regardless of the described setting.

CDS session governance

In order to be able to enforce various AAD policies, it is important to have the possibility to limit the session duration within CDS applications. Without any specific setting, the Power Platform session policy depends on the general settings within the AAD, which can lead to very long session durations that can go up to 90 days. In order to change this for CDS environments, we can define specific session timeouts by overriding the general AAD settings. The following session timeout types can be configured individually in every CDS environment in the environment settings (in the *Privacy+Security* area):

- **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 the own AAD and collaborate with the organization in specified areas. This capability makes it possible to give external users permissions to use certain Power Platform components such as Canvas Apps or Power Automate flows. In order to use this capability, the own AAD must allow inviting guest and every invited guest must first accept the received invitation in order to be physically registered in the AAD tenant.

Authenticating external users

All Power Platform components use AAD as the ultimate authentication provider, but with one exception. The exception is the **Power Apps Portals** component. This component provides a public, customer-facing portal capability that enables access to certain Power Platform content for the wider external audience.

Power Apps portals allow personalized access to customer-owned data such as customer profiles, customer service requests, customer field service work orders in CDS, to Power BI content or to SharePoint content. To achieve this, it is necessary to provide an authentication mechanism for external users, such as customers, partners, citizens, and others. There are two basic types of portal authentication:

- **Local authentication:** The user records with the authentication artifacts (username and password) of the external users are stored in the CDS Contact entity. 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 Apps portal.

- **External authentication:** The external user records are still stored and managed in the CDS Contact entity, but the authentication is offloaded to an external authentication provider and the authentication artifacts are not stored in CDS. This authentication type is disabled by default and needs additional configuration to be enabled.

The following portal authentication types are available:

- **Local sign-in:** Local authentication with the username/password stored in the Contact entity
- **AAD:** Enables authenticating internal organization users to the portal using their AAD credentials
- **AAD 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 the external users can select their favorite provider for registration and logging in. The following screenshot illustrates this capability:

The screenshot shows a registration form for a Power Apps portal. At the top, there's a navigation bar with a 'Home' link, a 'Sign in' button, a 'Register' button (which is highlighted with a dashed border), and a 'Redeem invitation' link. Below the navigation is a form with four required fields: 'Email *', 'Username *', 'Password *', and 'Confirm password *'. To the right of the 'Email' field are two authentication provider buttons: 'Azure AD' and 'Microsoft'. At the bottom of the form is a large blue 'Register' button. At the very bottom of the page is a dark footer bar with the text 'Copyright © 2020. All rights reserved.'

Figure 7.6 - Example portal authentication configuration

As you can see, the Power Apps 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, or Outlook) of external users.

The portal can either be configured for open registrations, which makes it possible for everybody to register, or can require certain invitation codes to be provided by the external users in order to be able to register.

An additional security option for Power Apps 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, it is necessary to have the capability to manage the privileges within the respective service. The Power Platform solution components provide certain authorization frameworks. In this section, you will learn about the authorization concepts for CDS, Power Apps, Power Automate, Power BI, and Power Apps portals.

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:

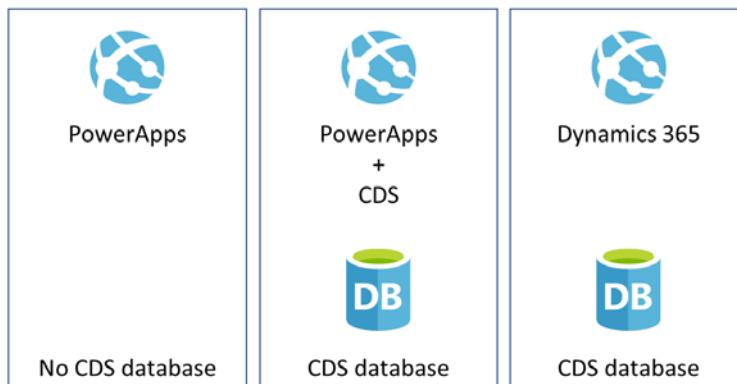


Figure 7.7 - Authorization – relevant environment types

As we can see, there can be a Power Platform environment without a CDS database, with a CDS database but without any Dynamics 365 app, and an environment with some Dynamics 365 apps installed on it. Environments without a CDS contain only built-in roles, which cannot be modified in terms of permissions. When we create a CDS database within an environment, security roles management switches over to the CDS database. Here, 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 security roles are available in the three environment types:

- **Type 1: Environment without CDS:**

Environment Administrator: This role grants full privileges in the environment.

Environment Maker: This role grants application-creating privileges in the environment.

- **Type 2: Environment with CDS:**

System Administrator: This role grants full permission in the environment, such as the Environment Administrator role in environments without CDS. If there were previous members in the Environment Administrator role, they will be moved to this role when a CDS 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 CDS artifacts.

Environment Maker: This role grants application-creating privileges in the environment, similar to the same role in environments without CDS.

CDS User: This is the basic user role with standard access privileges to business data.

Delegate: This role grants a 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.

Authorization in CDS and model-driven apps

The authorization concept of CDS-based applications is very complex and consists of several authorization possibilities, of which only a part is 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 CDS authorization.

Basics of CDS authorization

In order to understand the CDS authorization concept, certain basics need to be explained: **users and teams**, **business units**, record **ownership**, and **security roles** and **permissions**.

In the subsequent sections, we will take you through the details of these main components of CDS authorization.

Users and teams

Users and teams are the main security principals in a CDS solution, and the data about users and teams is stored in the corresponding CDS entities. 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 CDS solution, while teams do not have this capability.

The following types of users are available in CDS:

- **Read-Write:** A read-write user type is a standard type for normal interactive users using a CDS 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. They are created in the first step directly in AAD, without a license assignment. Then, in the second step, they are created in CDS by providing an AAD Application ID.
- **Stub:** A stub user type does not have the ability to log in and access CDS. This user type is only used for importing legacy data into CDS when there is a need to preserve the original record ownership of former users. The stub users can only be imported into CDS using an *Excel-based import* process.

The following types of teams are available in CDS:

- **Business unit-based owner teams:** This type of team is automatically created for every business unit that's created in the CDS. The membership in these teams is managed automatically by the CDS 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 CDS database contains a single business unit record immediately after creation. This represents the whole organization. The business unit entity 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.

Record ownership

Every record in a business entity in the CDS database, apart from some internal system entities, can have one of the following types of ownership:

- **Organization ownership:** Records in these entities do not have an owner; they belong to the whole organization and access management is simple: there is either access to all or to no records in the entity.
- **User or team ownership:** Records in these entities 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 entities, as well as entity-independent permissions and privileges.

The permission levels are illustrated in the following screenshot:



Figure 7.8 - CDS security role-based permissions

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 entity or to the respective functional capability.
- **User:** This permission level grants access only to your 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 entity, regardless of the owner, or the respective functional capability.

For entity-based permissions, there are two types of permission levels, depending on the entity ownership type:

- **Organization owned:** For organization owned entities, there can be only a permission type of **None Selected** or **Organization**. **None Selected** represents no access to records in the entity, while **Organization** represents access to all the records in the entity.
- **User and Team owned:** For user- or team-owned records, all five permission levels can be specified. The impact of the selected level on the access to records is explained in the next section of this chapter.

For entity-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 entity.

The types of transactions for which the permission can be specified for are illustrated in the following screenshot:

Create	Read	Write	Delete	Append	Append To	Assign	Share
--------	------	-------	--------	--------	-----------	--------	-------

Figure 7.9 - CDS security roles – entity transaction types

As we can see, the following transaction types can be permission-specified:

- **Create:** Permission is granted to create records in this entity.
- **Read:** Permission is granted to read records in this entity. When this permission is set to **None Selected**, the user does not see any records of this entity in the CDS application.

- **Write:** Permission is granted to modify existing records in this entity.
- **Delete:** Permission is granted to delete records in this entity.
- **Append:** Permission is granted to attach records from this entity to records in other entities.
- **Append To:** Permission is granted to attach records from other entities to records in this entity.
- **Assign:** Permission is granted to assign ownership of records in this entity to other user or team.
- **Share:** Permission is granted to share records from this entity with other users or teams.

The configuration of entity-independent permissions is much simpler and usually defines only a yes/no permission to perform certain actions in the CDS application.

Group teams

AAD, together with CDS, provides an alternative, simplified method of granting access rights within CDS 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 CDS team of the **AAD Security Group** or **AAD Office Group** type, depending on the AAD group type, and enter the **Object ID** you specified previously to create a relation to the AAD group.
3. Assign security roles to the CDS team.

After such a team is created, new users can be empowered to have appropriate access in the CDS application solely by adding those users as members to the AAD group in the AAD, without the need to manually assign them security roles inside CDS.

Important note

This alternate approach does not consider the possible need to assign a new user to a particular business unit. To ensure this process runs smoothly, the user must first be assigned to the business unit and only after that can they be added as a member to the AAD group.

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. CDS and model-driven apps have a one-to-many relationship, so within a single Power Platform environment with the CDS database, there can be many individual model-driven apps created. In order to manage the access to the apps, there is a capability to assign security roles individually to each model-driven app, as illustrated in the following screenshot:

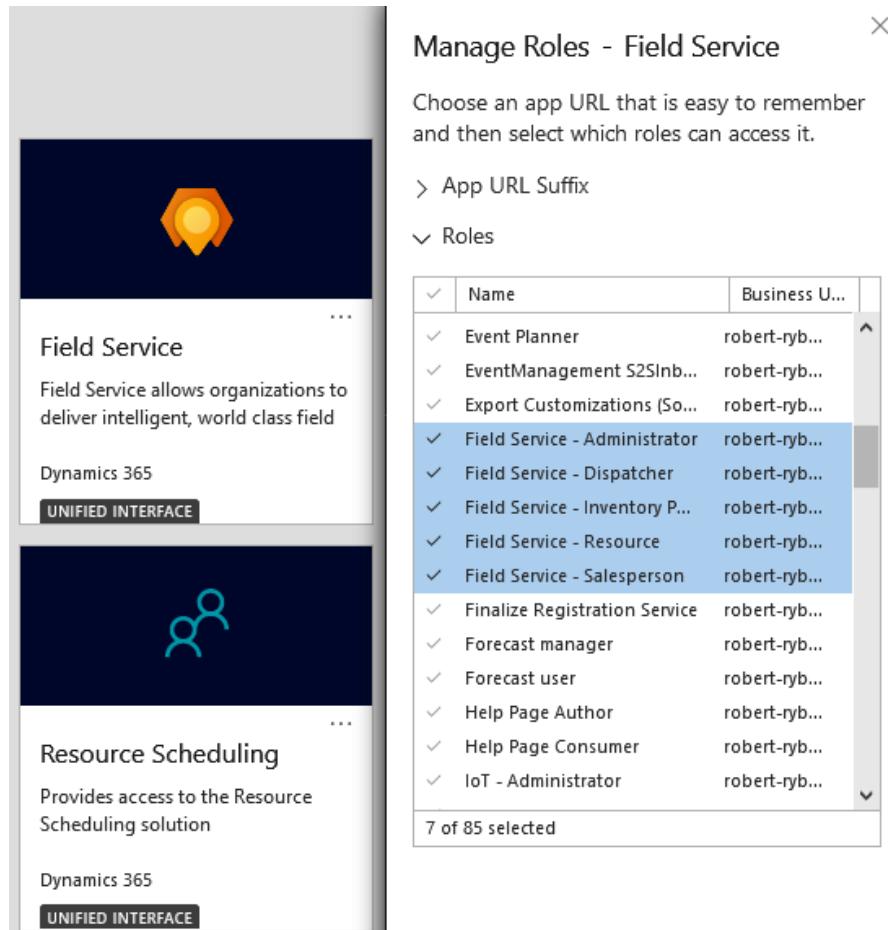


Figure 7.10 - Assigning security roles to model-driven apps

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, which are visible and available to users to apps just relevant to them.

Important note

A CDS system administrator always has access to all existing model-driven apps in a Power Platform environment.

Standard role-based security

Standard role-based security is mandatory and must be configured for every CDS application to allow access to data and business capabilities for the users. This security model makes use of the user's 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:

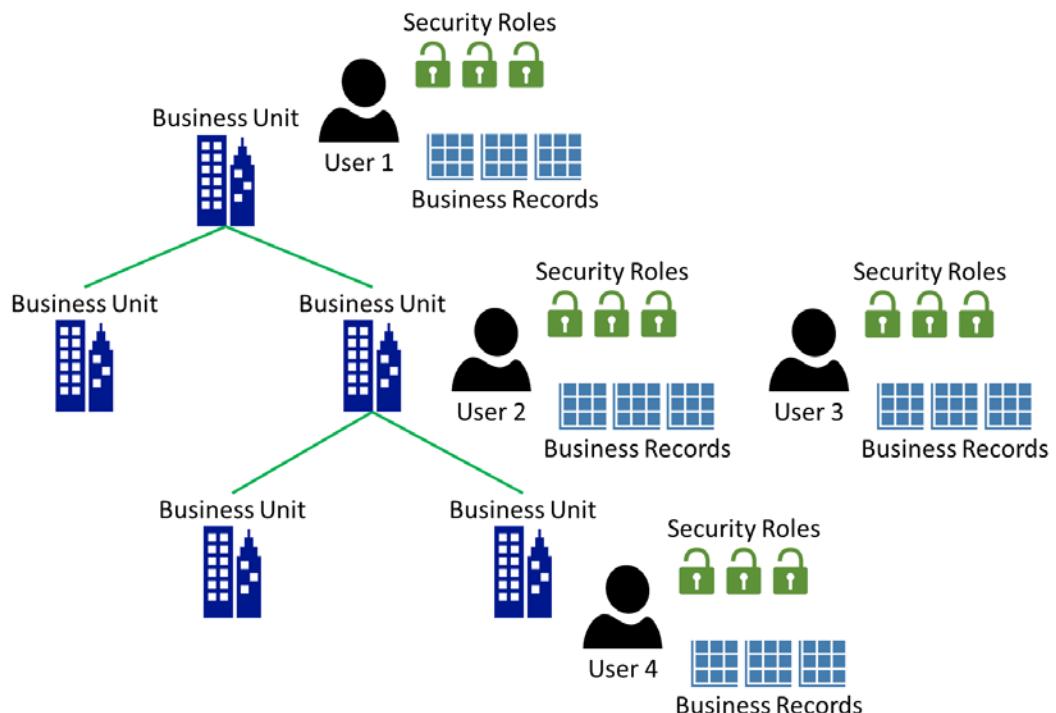


Figure 7.11 - Standard role-based security

As we can see, there is a hierarchical structure of business units, where the users are assigned to the business units and every user has one or more security roles assigned. Every user is an owner of certain business records in various entities. The authorization to see and manage data and to perform certain actions in the CDS application depends on the following:

- The level in the business unit's hierarchy that a user is assigned to
- The assigned security roles
- The ownership of records by the user and other users in the hierarchy

Let us explain this capability on an example using the previous diagram for the **User 2** considering only the transaction type **Read** for a certain business entity:

- **Option 1:** The user has the **None Selected** permission level in their security role for the entity. In such a case, the user will have no access to any records in the entity.
- **Option 2:** The user has the **User** permission level in their security role for the entity. 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 entity. 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 entity. 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 entity. 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.

In the following sections, you will learn about the additional, optional authorization options within a CDS-based solution.

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 CDS user can have another CDS 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

Important note

The hierarchy security capability can use either the manager hierarchy or position hierarchy, but not both at the same time.

In the following sections, you will learn about two options to grant ad hoc access to records.

Record sharing

Record sharing is, together with standard role-based security, the oldest security model in CDS 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 CDS 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 model*.

Access teams

The **access teams**-based security model is a good modern alternative to record sharing. This approach makes it possible to provide ad hoc record-based temporary access rights to individual users or teams. To set-up access teams capability for a certain entity, following steps are necessary:

1. Enable the access teams capability for a business entity.
2. Create an access teams template for the entity, defining the required privileges (read, write, delete, assign, share, append, and append to).
3. Place an access teams sub-grid on the entity form.

After finishing these steps, every user having full access to the record can manually add another user to the access teams sub-grid and so grant the specified privileges for that record to the user. The CDS platform autocreates an access team for the record after the first user is added to enable the sharing capability. Deleting a user from the sub-grid 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 CDS database and so has minimal performance impact on the CDS.
- 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 CDS database. In the next section, you will learn about the possibility to protect data at the individual field level.

Field-level security

There are often requirements to specifically protect individual fields in the CDS records, including highly sensitive fields such as usernames, passwords, and banking account or credit card details. Protecting individual fields can be implemented using **field-level security**. To set up this capability for a certain field, follow these steps:

1. Enable **field-level security** for a field in an entity.
2. Create field security profiles to specify field permissions (read, create, modify) for the protected fields.
3. Add users or teams as members to the field security profiles to grant the appropriate access.

After completing these steps, users not assigned to any field security profile will not have access to the protected fields in any part of the CDS application. This also applies to the CDS API, which also protects the content of those fields based on the user credentials used in the interface.

User interface security

The roles-based security model of using security roles also applies to certain user interface components in the CDS:

- **Entity forms:** For entity forms, there is a possibility to switch role-based security on in order to enable/disable access to the entity forms for various user groups based on security roles. This capability is available only for the *main* entity forms. It is important to always specify one entity form as a *fallback* so that every user always has at least one entity form available to work with. This capability makes it possible to provide the best user experience to various user groups by proving them tailored entity forms with just those data relevant for their job roles..
- **Dashboards:** For all types of dashboards (standard, interactive, entity dashboards, and so on), 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 entity-specific dashboards so that a dashboard is always visualized on the entity form.

In the previous sections, we covered all the authorization possibilities of CDS-based applications. Next, you will learn about authorization within Canvas Apps.

Understanding authorization in Canvas Apps

Canvas Apps are very different from CDS apps in terms of authorization. For Canvas Apps, the authorization can be divided into the following areas: authorization of apps and the authorization of connectors. 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 and that user is at the same time the owner of the app. In order to enable the use of the app to other users, 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**, which can be enabled in the AAD tenant. 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 connectors

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 connectors. Since there are more than 350 connectors and the number is still growing, there is no standard way to achieve authorization.

For the CDS connector, for example, there is the possibility to assign appropriate CDS 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.

Understanding 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 flows, 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 connector's authorization.

The flow author must provide proper authorization for all connectors used in the flow so that the flow has permission to perform the required transactions in the connected systems. Approval flows interact with end users since they expect approvals to be granted by the assigned approvers, but for this specific case, the approvers do not need any specific permission, just a Power Automate license to have access to the Power Automate portal. 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.

A specific category of background flows is **UI flows**. Since the UI 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 presented to the automated systems.

Authorization of interactive flows

The only interactive Power Automate flow type is the button flow. Making a button flow available to another user 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 connectors 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.

Understanding 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.

It is important to understand the dataset and query types used in Power BI. 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. 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, 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 in the underlying data source is honored.

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 role for a CDS dataset:

Manage roles

The screenshot shows the 'Manage roles' section in Power BI. On the left, under 'Roles', there is a list with 'Seattle' selected. Below it are 'Create' and 'Delete' buttons. In the center, under 'Tables', there is a list of entities: account, contact, incident, and opportunity. To the right, a 'Table filter DAX expression' field contains the formula `[address1_city] = "Seattle"`. There are also checkmark and delete buttons above the expression field.

Figure 7.12 - Power BI role-level security example

As you can see, the role "Seattle" is filtering all accounts and contacts in the CDS to those where the city = "Seattle". You will learn more details about Power BI authorization in the later sections of this chapter.

Understanding authorization in Power Apps portals

Power Apps portals is a Power Platform component that exposes certain portal capabilities and surfacing certain CDS data to an external audience – customers, partners, and citizens. It is necessary to be able to configure the following:

- Which content (navigational areas) will be available to anonymous public users
- Which content (navigational areas) will be available only to authenticated users
- Which CDS data types (entities) and individual records will be available only to authenticated users

These content- and data-based privileges are managed by a structure of permission-related settings, as illustrated in the following diagram:

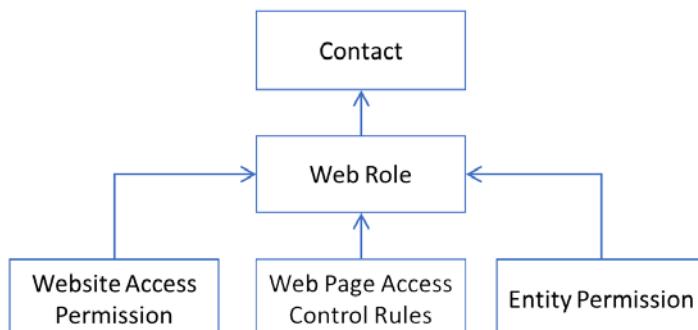


Figure 7.13 - Power Apps portal authentication structure

As we can see, the security principal for Power Apps portals is always the **Contact** entity. The main authorization entity in the structure is the **Web Role**; however, the actual permissions are specified in the other three entities. The following are the roles of the entities according to the previous diagram:

- **Contact:** Contact record represents the actual user of the portal. Every contact can be assigned one or more web roles.
- **Web Role:** Web Role is the main container for all privileges-based settings. Every web role can be assigned one or more website access permissions, web page access controls rules, and entity permissions.
- **Website Access Permission:** Website Access Permission specifies advanced frontend editing permissions for the whole Power Apps portals instance, specifically to edit content snippets, site markers, and web links on the portal. It is not needed to define these permissions for usual portal users.
- **Web Page Access Controls Rules:** Web Page Access Controls Rules specify permissions for individual web pages within the Power App 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.
- **Entity Permission:** Entity Permission specify detailed permissions on the level of the CDS entity, which means it has certain similarities to the CDS security role. 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 entity records (read, write, create, delete, append, and append to).

Important note

The Power Apps portal templates (customer self-service portal, partner portal, and so on) come with a predefined rich set of authorization settings that can be reused or modified. For a blank portal created from scratch, the preceding settings must be fully configured from scratch.

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 CDS, 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 the 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 the 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 the environments to fulfill the possible data residency requirements.
- Power Platform data, specifically data within CDS 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 CDS databases and metadata in Power BI, is encrypted at rest using SQL Server **Transparent Data Encryption (TDE)**. Normally, 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 CDS database. The own key is stored within Azure Key Vault for enhanced protection.
- Power BI data for **import** datasets and queries is stored in Azure Blob Storage and encrypted with encryption keys managed in an 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 **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.

CDS security roles

In this section, we will present some proven best practices for designing, implementing, and using CDS security roles.

Modifying security roles

Security configuration in CDS applications can be very simple, where the default CDS or Dynamics 365 security roles can be used without any modification. However, if the security requirements are specific and detailed, there might be a need to tailor the standard permissions. 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.

Security roles layering

Every user in a CDS-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 cause maintaining the solution to be challenging. An alternate 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 role layering, as illustrated in the following diagram:

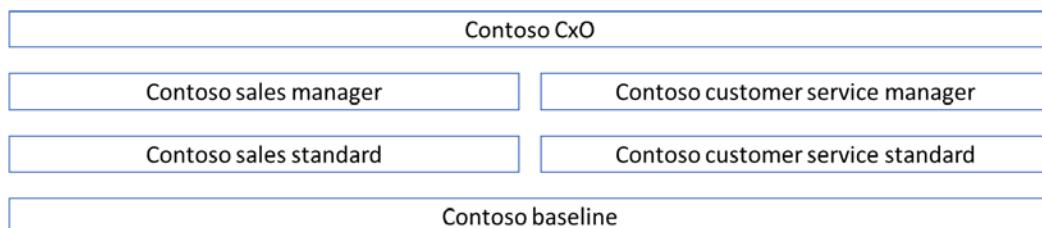


Figure 7.14 - Security role layering

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 benefits 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 and not all or many of them, as would be necessary when using the standard security roles approach.
- 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.

CDS content-based security

All the CDS security models described in this chapter are user-centric, for example the privileges granted are always inherited from the user accessing CDS. There are, however, situations where content-based security could be required. A typical example could be a CDS solution for a bank, where all private customer records are stored in the standard Contact entity. Let's assume 98% of the private customers are usual customers and 2% are VIP customers, which data should have a higher level of protection

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, leveraging only standard CDS 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 not accessible to the broad majority of the CDS application users. Access to those records could be managed using team membership or access teams. The benefit of this approach is that it is using standard CDS capabilities and that access to the protected data is reliably restricted in all parts of the CDS solution.

Using entity form switching

Using entity form switching is a frontend-only solution that leverages the capability of having more than one entity form per entity. Instead of managing access to entity forms based on security roles, this approach would switch entity forms based on content during the onload event. This approach does not protect all data, just selected data that will simply not be visible on the respective entity forms. This approach does require client-side custom development.

Note

A skilled CDS user would, however, be able to figure out the protected data using CDS capabilities such as advanced find, so all those additional CDS capabilities would need to be restricted as well.

Using server-side event handlers

Using server-side event handlers is a backend solution that leverages the capability of creating *CDS PlugIns*. 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** CDS events to protect the sensitive data in both entity views as well as entity forms. This approach requires custom development and protects the sensitive data reliably across all parts of the CDS solution, with the exception of the standard CDS 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.

Learning to 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 CDS business records. The best example of such a solution would be a CDS solution that's been extended with SharePoint for integrated document management and Power BI for analytics and reporting, as shown in the following diagram:

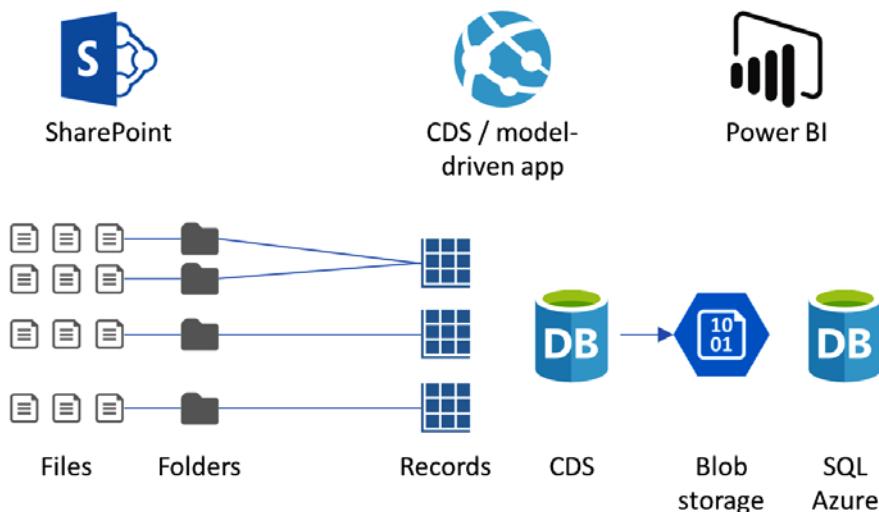


Figure 7.15 - Power Platform solution example

As we can see, the standard integration approach works as follows:

- For every SharePoint-enabled CDS entity, there can be one or more SharePoint folders where users can upload files and work with them from within the context of the CDS record in the **CDS** application. For this standard integration to work, every CDS user would need to have proper permission on the whole **SharePoint** site collection, integrated with the **CDS** application.
- A standard Power BI solution for **CDS** performs an import of the selected CDS records into the Power BI internal storage (**Azure Blob Storage** and **Azure SQL database**) and uses this local copy of the CDS 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 CDS entity.

Both of these standard solution approaches do not respect the very detailed authorization concept of the CDS 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 CDS application user would be able to switch over to the integrated SharePoint site collection and find any files belonging to any CDS records they have no permission to see within CDS.
- Every CDS application user would see all the CDS 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 CDS-based access permissions given 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.

CDS-SharePoint integrated security

Achieving a consistent and automated authorization solution for CDS and SharePoint is possible using some of the following options:

- **Full access:** The standard setup for CDS-SharePoint integration requires giving respective contributor permissions to the whole SharePoint site collection, which is used for integrating with CDS. 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 to grant site collection wide permission to all CDS users, but rather grant access to individual folders and files in the SharePoint site collection hierarchy manually by an administrator. While this is theoretically possible, for a larger number of CDS business records, this would be impossible to manage.
- **Permission replication development:** This approach would require building a custom solution for CDS to replicate the CDS record permissions in the integrated SharePoint folders and files. This would require significant custom development effort to cover all the very specific CDS authorization possibilities and their changes over time.
- **Third-party solutions:** There are commercial solutions on the third-party market that cover permission replication from CDS to SharePoint.

CDS-Power BI integrated security

When integrating Power BI with CDS, 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 CDS 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 entities 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 CDS 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 CDS 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 CDS authorization. This authorization can be implemented using both *import* as well as *DirectQuery datasets*. The configuration would require using credentials for the datasets allowing full access to all data in all entities used in the Power BI visuals together with creating of the respective RLS roles and assigning them to Power BI users.
- **CDS 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 CDS 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 CDS.

Important note

DirectQuery for CDS is currently a preview feature and is not recommended to be used in production scenarios.

In the following sections, we are again discussing the topic of automating the whole identity and access management, this time with including the possible integration to 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's 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 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:

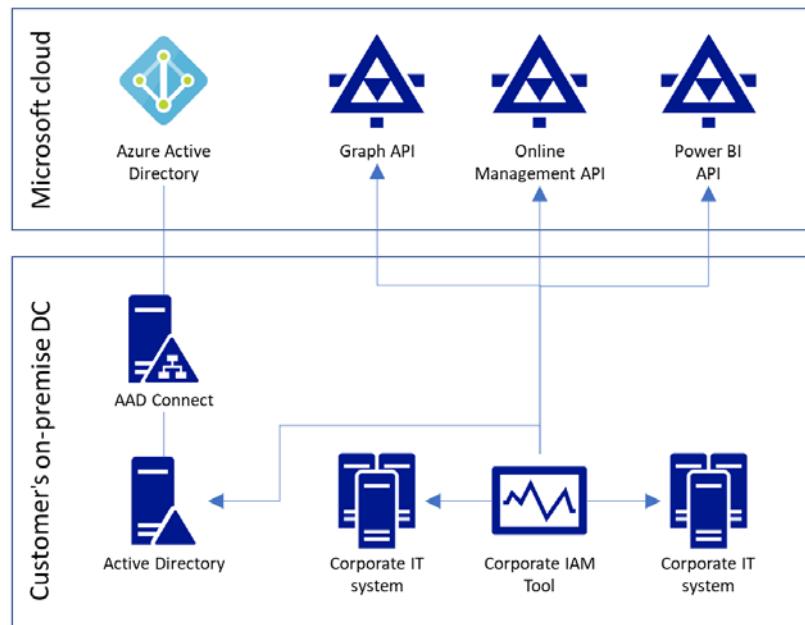


Figure 7.16 - Identity and access automation

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, following are the necessary 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*, whether the new user account has already been provisioned in AAD.
3. OPTIONAL: After the new user account has been provisioned in AAD, assign the user to all necessary Office 365 groups associated with the CDS 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 CDS environments is also an asynchronous process, start a polling sub-process to poll all the CDS environments, whether the new user account has already been provisioned.
6. In every CDS environment where the new user is already provisioned, assign the necessary authorization privileges and permissions using the Online Management 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:

Azure Active Directory Tenant

- Conditional Access
- Active Directory Integration
- Multi-Factor Authentication
- Data Loss Prevention Policies
- Cross-Tenant Inbound and Outbound Restrictions

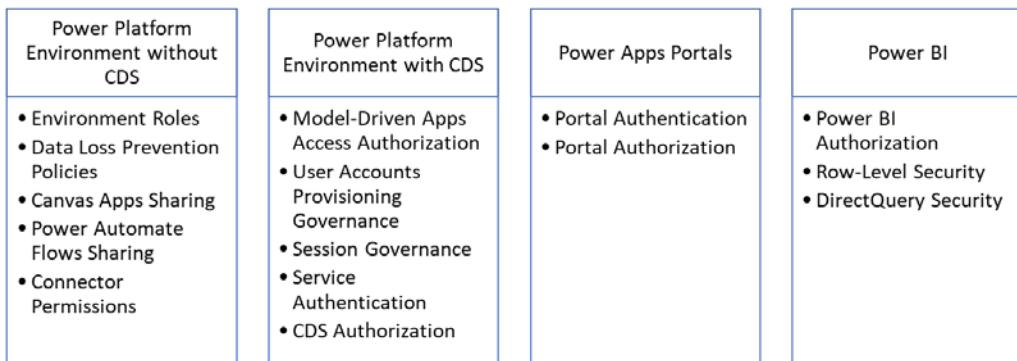


Figure 7.17 - Mature Power Platform security model

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, 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 CDS environments.
- **Session governance:** Use session governance to prevent the misuse of CDS applications by unauthorized users and to enforce changed policies, which are applied during the user authentication process.
- **Service authentication:** Use the proper service authentication model when developing external applications that connect to CDS or Power BI environments.
- **CDS authorization:** Plan, design, and implement a proper CDS authorization model to ensure every user group has adequate permissions within CDS 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, the 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 to fully block some social connectors and include only the business-relevant connectors in the Business Data group.

Common Data Service

For CDS, model-driven apps, and Dynamics 365, Contoso Inc. decided on the following security principles:

- For every new CDS-based environment, an 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 CDS security roles, to avoid providing an unnecessarily high number of not needed apps to standard users.
- Session timeouts and inactivity timeouts will be made mandatory for every new CDS environment.

- For every new CDS environment, a proper authorization model will be configured, using strictly only the standard CDS authorization features. Using the record sharing feature will be not permitted.
- Any integration with a CDS-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 multi-factor authentication.

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 travelling 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 to leverage the integrated authorization capability of this approach.

Contoso Inc. will also consider using a third-party solution to replicate permissions from their CDS-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 the Power Platform security, authentication, and authorization, the different types of identities, and the integration options available within an on-premises **Active Directory**. You have seen, what are the possibilities for authentication of external users. Further you have 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.

In the next chapter, we will focus on the extensibility options of the Power Platform solution components.

8

Microsoft Power Platform Extensibility

In this chapter, you will learn about the customization and extensibility options available for Power Platform components. As discussed in *Chapter 4, Tools and Techniques*, the configuration and customization can be usually performed by a citizen developer, while for custom development an IT pro developer is required. 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 use best practices for Power Platform extensibility.

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

- Extensibility overview
- **Common Data Service (CDS)** and model-driven apps 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 family to customers, making it possible for them 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:

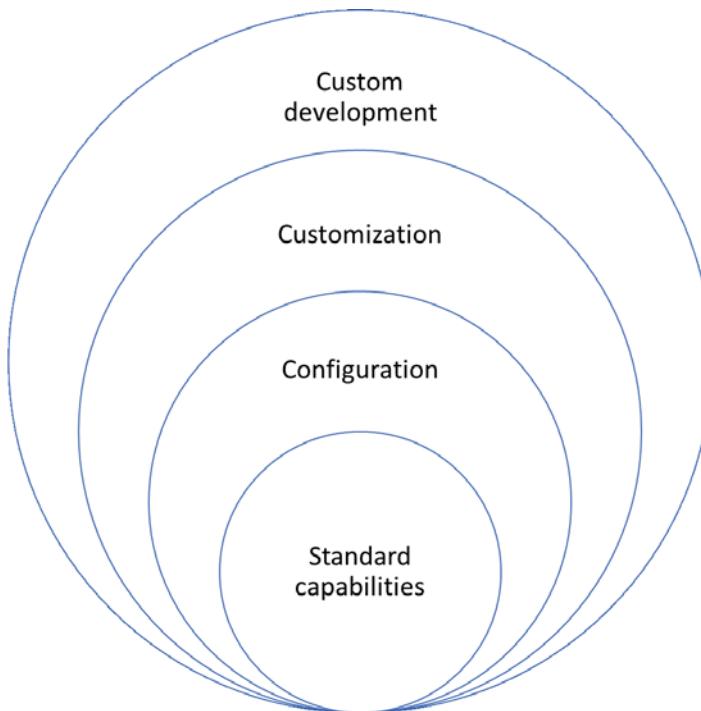


Figure 8.1 – Power Platform extensibility options

As we can see, the options are as follows:

- **Standard capabilities:** Some of the Power Platform components can be theoretically used as they are, and this is true specifically for Dynamics 365 apps, which are already implementing 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. Using standard capabilities does not require any effort to enable them.
- **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 to define the product catalog in Dynamics 365 Sales or to design marketing emails in Dynamics 365 Marketing. Configuration, in most cases, can be 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 CDS 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, the citizen developer skills are sufficient. This can be either 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 PlugIns 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 first start with the CDS and model-driven apps extensibility.

Presenting CDS and model-driven apps extensibility

CDS 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 sever-side custom development*.

In the following sections, we will explore standard CDS customization, CDS automations, as well as possibilities for custom development. At the end of this section, you will learn about the extensibility of two specific CDS-related components: Power Apps Portals and Unified Service Desk.

Understanding CDS standard customization

CDS standard customization encompasses a lot of different parts of the structure of CDS, 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 CDS and model-driven applications.

In every Power Platform environment, there can be only one CDS instance, and the whole customization is performed within this one instance of CDS. It is, however, possible to create multiple model-driven applications, and every such application can provide a subset of the structure of the CDS to the end user so that the relationship between CDS 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 CDS solution.

The relationship between model-driven applications and CDS can be illustrated with the following diagram:

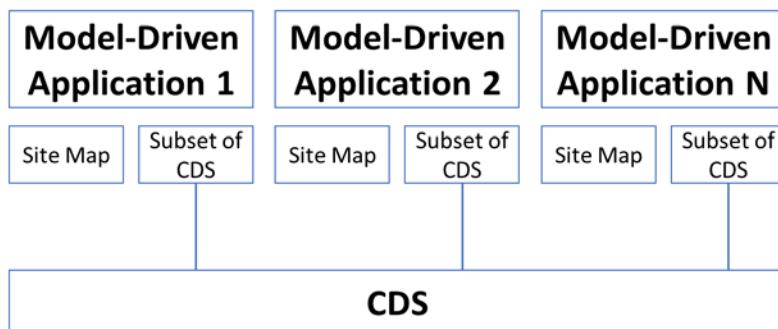


Figure 8.2 – Relationship between model-driven apps and CDS

As we can see, every model-driven app consists of the following main parts:

- **Site map:** Site map contains the main application navigation structure.
- **Subset of CDS:** Subset of CDS contains a collection of references to selected CDS artifacts.

Which artifacts can be selected, and which are included automatically, can be explained using the following simplified diagram:

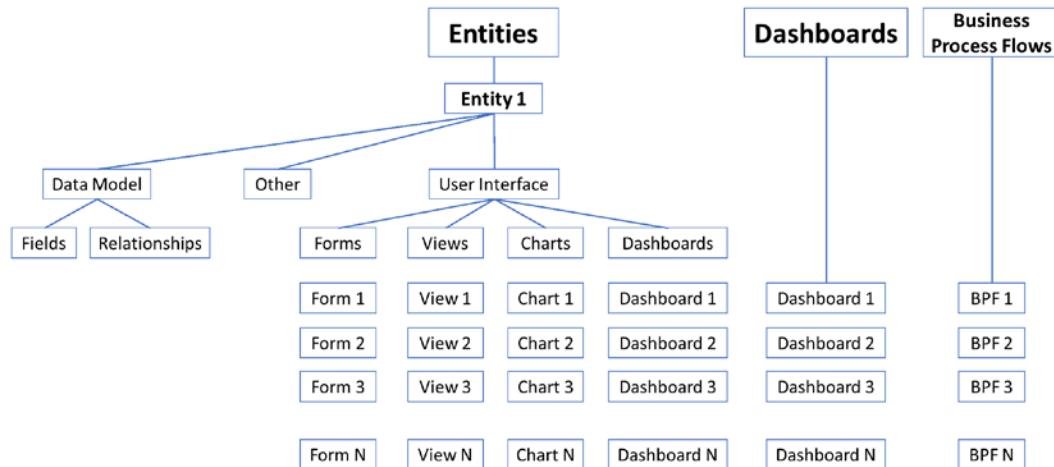


Figure 8.3 – Selectable artifacts for model-driven apps

The preceding diagram presents a very simplified structure for various CDS artifacts, just for the purpose of explaining the *model-driven apps* approach. When configuring a model-driven app, it is possible to select individual entities from the overall entity collection from CDS. When selected, the following artifacts of the entity are automatically included in the app:

- The whole **data model** of the entity, consisting of fields, relationships, and other artifacts
- **Other artifacts**, such as business rules, hierarchy settings, and alternate keys

There are also entity artifacts, which can be individually selected as part of the model-driven app:

- **Entity forms:** It is possible to select one or more existing entity forms.
- **Entity views:** It is possible to select one or more existing entity views.
- **Entity charts:** It is possible to select some of the existing entity charts. Entity charts are not mandatory, so there is no requirement to include any charts in the model-driven app.
- **Entity dashboards:** It is possible to select some of the existing **entity-specific** dashboards. Entity dashboards are not mandatory, so there is no requirement to include any dashboards in the model-driven app.

Beside these entity-specific artifacts, it is also possible to individually include the following entity-independent artifacts:

- **Dashboards:** It is possible to select some of the existing **entity-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 entity that is BPF-enabled and included in the model-driven app's site map.

Important note

It is necessary to include all entities used in the model-driven app's site map in the application itself.

The standard customization of CDS consists of certain basic areas:

- **CDS data modeling:** Used to design and implement the required data model
- **CDS user interface design:** Used to design and implement the standard user interface elements
- **CDS automations:** Used to design and implement standard CDS automations

The CDS standard customization is performed with the Maker Portal, as described in *Chapter 4, Tools and Techniques*.

CDS data modeling

CDS comes with a basic data model consisting of standard, general-purpose entities that are used in every relationship management solution, such as account, representing the company customers, and contact, representing the people to contact. The basic CDS 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 entities.
- Any **third-party CDS-based application** coming from an implementation partner or from AppSource extends the data model with all required entities, fields, and relationships.
- **Customer-specific standard customization** performed within a solution implementation project can also extend the data model accordingly.

The following are the possibilities to extend the CDS data model, regardless of the presence or absence of any Dynamics 365 or third-party CDS application already installed in the CDS environment:

- **Basic, standard, general-purpose CDS entities** cannot be deleted, and there are limited possibilities when it comes to modifying the standard entity fields or relationships. It is possible to extend the data model with custom fields and relationships. These custom artifacts can be modified or deleted at any time.
- Any entities coming from a **Dynamics 365** or a **third-party CDS-based application** within managed solutions cannot be deleted. There are limited possibilities when it comes to modifying the standard entity fields or relationships, as specified in the managed properties of these artifacts. It is possible to extend the data model with custom fields and relationships. These custom artifacts can also be modified or deleted at any time.
- It is possible for you to create your own **custom** entities with custom fields 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 entity, the following are the most important options to consider:

- **Entity type:** An entity can be a **standard** entity or an **activity** entity. An activity entity extends the existing group of activity entity types (email, appointment, phone call, letter, fax, and so on) in CDS with a new activity type. The new custom activity entity will be handled within the CDS solution's activity management like any other activity.

- **Entity ownership:** An entity 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 entity.
- **Other settings:** Specifying some additional entity behavior, such as the possibility for activity records to be assigned to the entity, to support emailing to the entity or to enable connections, access teams, and document management.

Important note

Certain entity settings cannot be modified after the entity has been created, so it is necessary to plan for those settings accordingly. Any incorrect settings would require that the entity be deleted and completely recreated.

When a custom entity is created, entity fields and relationships need to be created. When creating new fields, the following are the most important options to consider:

- **Field type:** There are three basic field types: **standard**, **calculated**, and **rollup**. The field type cannot be changed after the field has been created.
- **Data type:** There are several standard as well as CDS-specific data types available. The data type cannot be changed after the field has been created.
- **Required status:** This setting defines whether a value in the field is mandatory when creating or modifying records of the field. There are three possible values for this setting: required, recommended, and optional.
- **Other field settings:** There are several other important settings, such as whether the field 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 entity is created between the two related entities.

- **Type of behavior:** There are specific settings that define how the relationship will behave for the records on the *many* side of the relationship, for certain events happening 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 CDS 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 CDS user interface elements.

CDS user interface design

The visible part of the overall CDS 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 as well as to interact with the data by creating, modifying and deleting records, or triggering various functions.

In the following sections, you will learn about the most important information you need to know about user interface elements.

Entity forms

Entity forms are used within a model-driven app for creating, displaying, and modifying data in individual CDS entity records. The following entity form types are available:

- **Main Form:** The main form is used for creating, displaying, and modifying individual CDS entity records. There can be multiple main forms per CDS entity, and the main forms can be assigned to CDS security roles to be able to assign the best entity form to the appropriate user groups. Furthermore, a 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**, and **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 entities (from an *N:1* relationship) on the main form. The quick view form can contain fields from the related entity, as well as sub-grids displaying related data from the quick view form's entity itself. The data in the quick view form is always read-only, when embedded on the main form. The quick view form can only contain a limited number of control types. There can be multiple quick view forms per entity. 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 entity in that we only enter data into the most important fields. There can only be one quick create form per entity, and the quick create capability must be separately enabled for the respective entity. 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 on interactive experience dashboards. The form must have a specific design and can only contain a limited number of control types.

The next type of user interface element we will discuss is the entity view.

Entity views

Entity views are used within a model-driven app to display, but also modify, entity records in a table view. There are several view types available for a CDS entity. There are system view types such as **Advanced Find View**, **Associated View**, **Lookup View**, and **Quick Find View**, which are automatically generated when a new custom entity is created. New entity views can only be created using the **Public View** type. The following parameters can be configured when creating new, or modifying existing, entity views:

- Entity fields as columns, including their position and width
- Sorting records across multiple fields
- Filtering records using a comprehensive filter editor

It is possible to configure a CDS entity so that it supports alternate view types. Examples of such alternate visual representations of entity records are as follows:

- **Editable views:** This option makes it possible to update entity records directly in the entity view, without the need to open the record in an entity form.
- **Calendar views:** This option makes it possible to display the entity records on a calendar control. This option is suitable for entities that represent time entries with typical fields, such as *Time from* and *Time to*.
- Any other standard or custom-developed custom controls.

Entity views, compared to entity forms, cannot be assigned to security roles, so there is no way to configure user-specific entity view assignments. Public views, however, can be deactivated so that they're unavailable for all users.

Beside publicly available entity views, users can create, save, and eventually share their own personal views using the *Advanced Find* capability of CDS.

Next, we'll discuss entity charts.

Entity charts

Entity charts are graphical representations of data in a certain CDS entity. Entity charts can be visualized either as part of entity views, or by placing them on CDS 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 system charts, the following configuration possibilities are available to you:

- An entity chart is always entity-specific, so it can only visualize data from a single entity.
- There are different selectable types of charts; for example, column, bar, area, line, pie, funnel, tag, and doughnut.
- Field series for values (y-axis) with different aggregations (sum, average, count, min, and max) and field 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.

Entity charts have a similar behavior to entity views as they cannot be assigned to security roles, and there is also a possibility for end users to create their own personal charts.

Entity-specific and entity-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 entity views, entity charts, web resources, IFRAMES, and special components such as timeline, 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 their 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 entity streams and entity charts, and the data is filtered using visual filters.

Entity-independent dashboards are used as home screens in model-driven applications, while entity-specific dashboards are usually configured to be displayed instead of entity views or within entity forms.

Beside 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 where they have access to such dashboards. Furthermore, users can also include whole Power BI dashboards and save them as personal dashboards in CDS.

Important note

It is not in the scope of this book to dive deeper into the details of the standard CDS 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 CDS customization artifacts

There are some other CDS standard customization artifacts worth mentioning:

- **Hierarchy Settings:** This feature makes it possible to visualize records in a certain entity as a graphical hierarchy. In order to achieve this, it is necessary to perform certain specific configurations. You need to create or use a self-relationship within the entity, create a specific quick view form for visualizing the most important data, and configure the hierarchy setting within a standard CDS customization.
- **Global Option Sets:** Option sets is one of the specific CDS data types. There are two types of option sets – local option sets, created directly within an entity along with the values and global option sets, which are created outside of an entity. Global option sets can be reused in any entity for a field of type option set. The benefit of global option sets is consistency in values of the same type across multiple entities.
- **Reports:** For years, the traditional reporting capability for CDS was the Microsoft SQL Server Reporting Services technology. This capability is still used, and every CDS-based solution usually comes equipped with a set of standard reports. Simple reports can be created directly with a wizard from within the CDS environment. However, for complex reports, is it necessary to use a *custom development* approach. Generally, traditional reports are losing relevance within CDS 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 CDS 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 CDS solution to end users. The steps for creating a model-driven app are as follows:

1. Have all the necessary CDS 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 main app's navigation.
All the entities that are selected in the site map are automatically added to the app's configuration.

4. Select the app's artifacts (entity-independent dashboards; business process flows; entities with selected forms, views, and charts; and entity-specific dashboards).
5. Save and publish the app.
6. Assign proper CDS 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 CDS 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, tablets) when using the mobile 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 entity main forms, the tabs, sections, individual fields, sub-grids, 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.

Understanding CDS automation

CDS supports multiple automation possibilities to simplify and consolidate certain processes, to make data entry easier, and to make performing certain processes entirely automatic. All the standard automations are sometimes referenced as **Processes**. In the following sections, we will look into various automation approaches.

CDS business rules

CDS business rules is the simplest automation option within CDS. The purpose of business rules is to provide certain automations on the client-side or on the server-side within a single CDS record. A business rule consists of a set of conditions and actions, as illustrated in the following diagram:

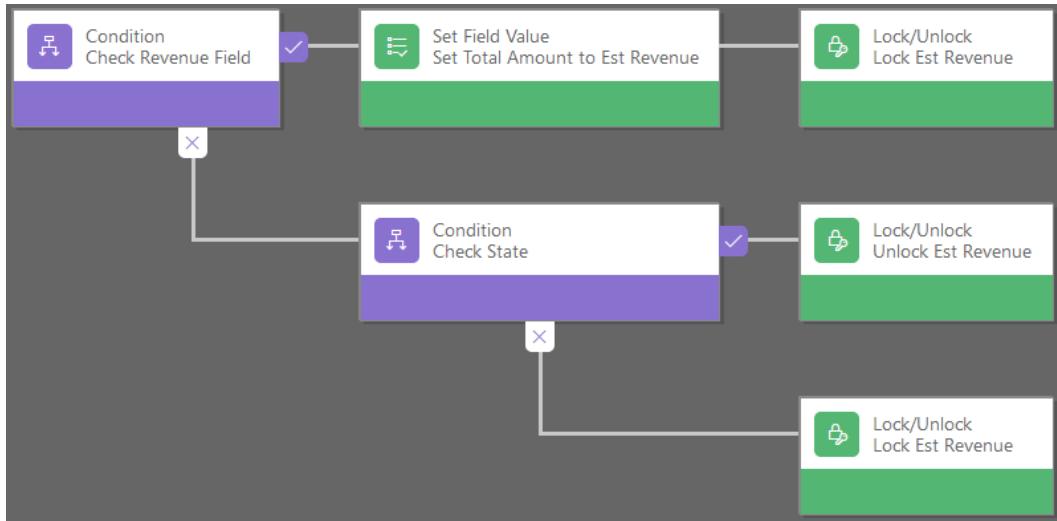


Figure 8.4 – CDS business rule example

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 field
- Set the value of a field
- Lock/unlock a field on the entity form
- Make the field visible/invisible on the entity form
- Set the field as business required or optional on the entity form
- Show an error message based on a condition
- Provide a recommendation based on a condition

Business rules are created within the scope of a certain entity, and all conditions and actions always refer to the values of a record in that entity.

CDS workflows

CDS workflows is a traditional automation approach that comes from the early days of Microsoft Dynamics CRM. The purpose of workflows is to perform certain automations within CDS in a synchronous or asynchronous way. A CDS workflow can be triggered by an event coming from a CDS record (record created, updated, or deleted), can be triggered from within a **Business Process Flow (BPF)**, or can be triggered on-demand by a CDS user. A CDS 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)
- CDS record transactions (create record or update record)
- Trigger a custom workflow action (a form of custom-developed business logic, described later in this chapter)

Important note

The possibilities that are available when using traditional CDS 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. Microsoft provides this recommendation directly in the CDS workflow designer.

CDS custom actions are very similar to CDS workflows. We'll look at these in the next section.

CDS custom actions

The primary purpose of **CDS custom actions** is to provide an easily configurable piece of automation functionality that can be called from different places of a CDS application.

CDS custom actions are similar to CDS workflows but have the following differences:

- CDS custom actions do not have triggers; they must be triggered from outside by CDS workflows, BPF, or by code from within the CDS API.
- CDS custom actions allow us to define input and output parameters so that we can exchange values with the calling service.

Except for these differences, CDS custom actions provide the same business logic capabilities as CDS workflows.

CDS business process flows

CDS business process flows (BPFs) is 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:

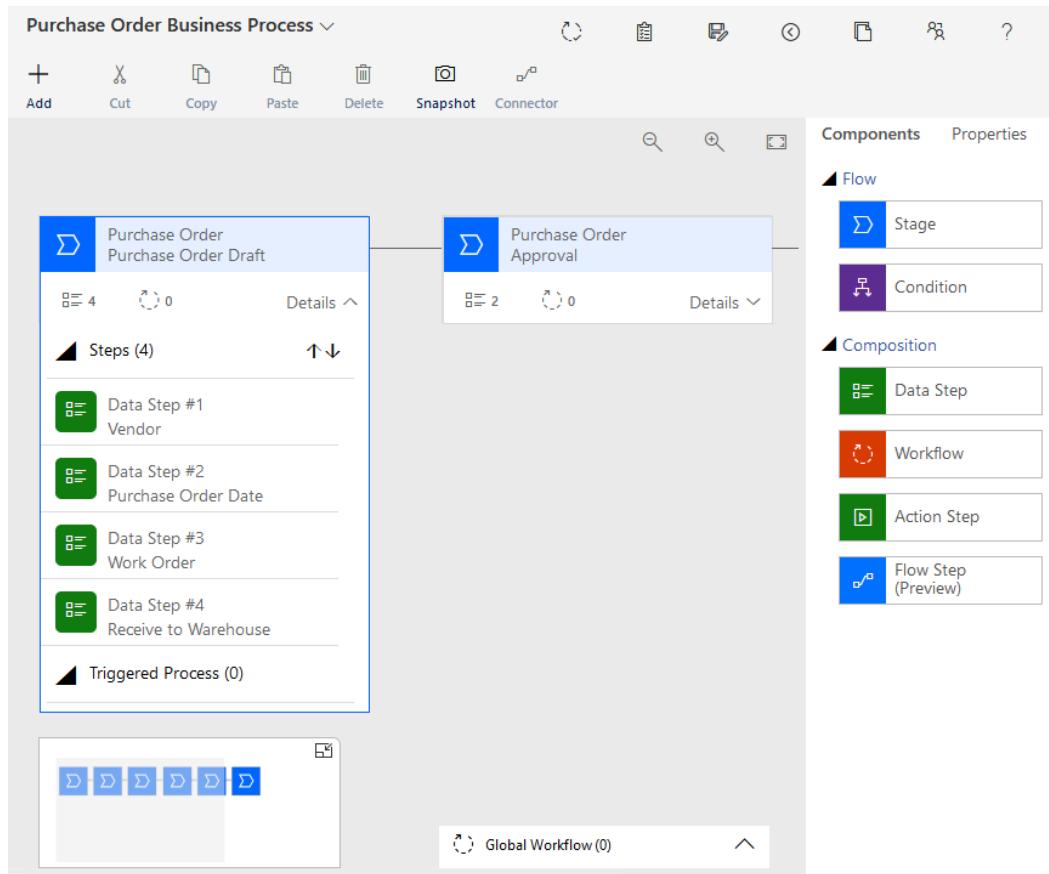


Figure 8.5 – Business process flows example

As we can see, BPF provide the following capabilities:

- A series of **stages** and **conditions** that guide the user through the main stages of a business process. The conditions can support implementing branching logic.
- In every stage, there can be a certain number of **data steps** for data entry, **actions steps**, or **flow steps**, all of 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 can be triggered automatically on stage-entry or stage-exit.
- In addition, we can configure **global**, stage-independent workflows, which can be triggered during major BPF 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 also have a specific feature that enables security by assigning security roles. By doing this, we can assign proper BPFs to every end user group. A BPF can encompass multiple entities, and there can be up to 10 different BPFs assigned in parallel to every CDS record.

Power Automate

Compared to the previously described automation options, **Power Automate** is independent from CDS. The automation flows can be implemented as cross-applications and don't necessarily need to use CDS. However, it is important to understand that Power Automate flows can be distributed using Power Platform solutions between environments that have already deployed a CDS. For Power Platform environments without CDS, the Power Automate flows can only be distributed individually using an export/import procedure.

Compared to CDS workflows, the capabilities of Power Automate flows are broader. Power Automate flows work with the concept of data connectors to connect to IT technologies, systems, and solutions. In order to automate processes in CDS-based applications, Power Automate makes use of the CDS connector for triggering a flow as well as performing actions in CDS. There are currently two different types of CDS connectors:

- **Common Data Service Connector:** This connector can be used for Power Automate flows that are created within a CDS solution, or outside of a CDS solution. For this connector, the trigger, as well as any action step in the flow, must be configured for a selected Power Platform environment. When deployed to a different Power Platform environment, the connection to the new environment might need to be manually updated.
- **Common Data Service (current environment) Connector:** This connector can only be used from within a CDS 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.

Both CDS connectors can trigger the flow by following events in CDS:

- **CDS record created:** For the CDS (current environment) connector, the trigger can be combined with update and delete.
- **CDS record updated:** For the CDS (current environment) connector, the trigger can be combined with create and delete.
- **CDS record deleted:** For the CDS (current environment) connector, the trigger can be combined with create and update.
- **CDS record selected:** This is an on-demand trigger. The flow is started manually by a CDS user from a selected CDS record. The trigger is not available for the CDS (current environment) connector.
- **BPF trigger:** This is an on-demand trigger. The flow is started manually by a CDS user from a BPF.

Compared to CDS workflows, however, CDS connectors offer many more **actions** when it comes to CDS data:

- **Create a CDS record:** This action creates a record in a CDS entity. This action is similar to what we can use in CDS workflows.
- **Update a CDS record:** This action updates a record in a CDS entity. This action is similar to what we can use in CDS workflows.
- **Delete a CDS record:** This action deletes a record in a CDS entity. There is no such action for CDS workflows.
- **Get a CDS record:** This action is not available for CDS workflows. This action makes it possible to retrieve a single record and use its values for the business logic.
- **List CDS records:** This action is not available for CDS workflows. This action makes it possible to retrieve a list of records and use its values for the business logic in a loop.

In addition, the CDS connector (current environment) offers the following actions:

- **Execute a changeset request:** This action can perform multiple 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 CDS record.

- **Perform a bound or unbound action:** This action can call a CDS 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 entity records with a 1:N relationship.

Important note

You might still see the CDS automation **Dialog** type available when creating a new process. However, Dialogs are deprecated and should not be used for new CDS implementations anymore.

With this, we have concluded the pure customization possibilities for CDS applications. In the following sections, you will learn how to extend CDS with code on the client side, as well as on the server side.

Understanding CDS client-side extensibility

In this section, you will learn about the various extensibility options that are available on the client side of a CDS-based solution.

CDS 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 CDS, specifically entity forms or views, is to use custom controls instead of the default controls. CDS 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 field-related standard custom controls are available for various field 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 entity form designer in the Maker Portal. For every eligible field data type, you can select some of the available custom controls and decide what device type (web, tablet, phone) this custom control will be used on. The customization possibility is illustrated in the following screenshot:

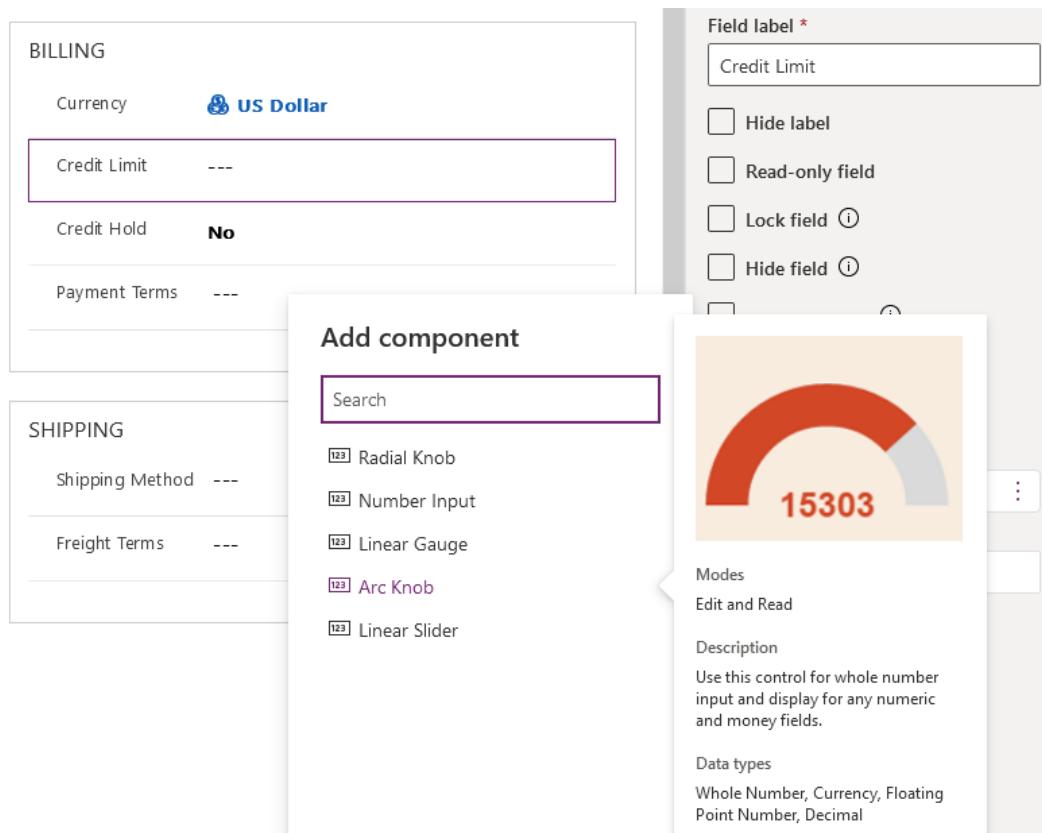


Figure 8.6 – Configuring custom controls

As you can see, for a numerical field, there are several custom controls that can be used on the entity form instead of a default textbox. The Maker Portal offers all available custom controls for the selected field 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 minimum and maximum values. If an entity field visual representation is replaced with a custom control, the same visual representation will be used within BPF in the data steps.

Standard custom controls can be also used to replace the entity views with other user interface elements. Standard entity views show the entity 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 **editable view**, **calendar view**, **timeline control**, and **Kanban control**.

As for entity fields, not every custom control can be used for every entity. The respective entity must have certain specific fields and every custom control requires specific configuration settings. It is possible to enable multiple custom controls for an entity, which gives the user the ability to switch the visual representation of the entity records.

Power Apps Component Framework

Power Platform allows users to develop own custom controls to be used on the CDS user interface (entity fields 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**.

Important note

Custom controls developed with PCF can be used for canvas apps the same way as they can for CDS apps. At the time of writing this book, this capability is in preview.

PCF code components are solution-aware and 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, 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 command-line interface 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 Apps 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 Apps 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 a new or existing CDS solution for use within CDS and for deployment purposes. This task also consists of several command-line commands. When the component is uploaded into CDS, it can be used for the entity fields or views the same way as standard custom controls, in order to replace the default graphical representation.

The benefits of using PCF code components to extend the CDS user interface are as follows:

- Using PCF code components is performance optimized by design, since they become part of the application rendering process.
- It is possible for the code components to interact with the CDS 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 CDS user.
- It is possible to use device-specific features such as a camera, microphone, and GPS location.
- The PCF components can be easily reused across a whole CDS 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 CDS web resources.

Web resources

Using web resources is the traditional approach of extending the CDS 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 CDS user interface (entity 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 CDS are basically different file types that are uploaded into the CDS database and used for different extensibility purposes. These web resources are solution-aware, can be uploaded into a CDS solution, and 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 CDS. 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 CDS. Typical representations are *PNG, JPG, GIF, ICO, and SVG* components.
- **Code web resources:** These web resource types are used to implement certain business logic within the user interface web resources, or directly within the CDS user interface. 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 CDS user interface with non-standard UI elements or even whole additional solutions.

Compared to custom controls, web resources are not used to replace the visual representation of CDS entity fields or views; rather, they are placed on the CDS application user interface (entity forms and dashboards) using a dedicated control type – a web resource. Only **form-enabled** web resources (HTML and graphical web resources) can be added to the CDS entity 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 CDS forms or dashboards. In order to achieve this, all the used files need to be uploaded as web resources into CDS. 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 CDS entity form elements and to communicate with the CDS 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 CDS user interface elements (CDS entity 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 entity forms. The following are the most important and frequently used:

- **Form OnLoad:** This event happens during the initial load of a CDS entity form. The event handlers registered here can perform initial manipulations of the entity 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 CDS entity form's controls, immediately before the form's data is saved.
- **Field OnChange:** This event happens after the user leaves a CDS entity form's field, in case the value of the field was changed. The event handler registered here can perform certain manipulations, depending on the current value of the particular field.

There are other events related to specific parts of a CDS entity form, such as sub-grids, IFrames, lookup controls, the knowledge base search control, and the entity 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 CDS entity form (tabs, section, fields, and other controls), performing calculations with data in the entity 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 CDS API method using a client-side Web API , or using some device-specific features on mobile devices (images, audio, video, 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 CDS business rules, there is a certain overlap since the purpose of CDS business rules is to minimize the use of code. However, at the time of writing, the possibilities that are covered by CDS business rules are rather limited, covering only the most frequently used types of client-side manipulations. There's still a lot of requirements that cannot be covered with CDS business rules and need to use form scripting.

Command bar/ribbon extensibility

Web resources of type code (JavaScript) 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.

Beside 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 entity type. This navigation is visually represented as the navigation bar with navigation elements (buttons) in the upper part of the entity views, forms and sub-grids, as illustrated in the following screenshot:

All Accounts					
	Account Name	Main Phone	Address 1: ...	Primary Contact	Email (Primary Contact)
	Fourth Coffee (sample)	555-0150	Renton	Yvonne McKay (sample)	someone_a@example. Active
	Litware, Inc	555-0151	Dallas	Susanna Stubberod (sampl	someone_b@example. Active
	Adventure Works (sample)	555-0152	Santa Cruz	Nancy Anderson (sample)	someone_c@example. Active
	Fabrikam, Inc. (sample)	555-0153	Lynnwood	Maria Campbell (sample)	someone_d@example. Active
	Blue Yonder Airlines (sample)	555-0154	Los Angeles	Sidney Higa (sample)	someone_e@example. Active

Figure 8.7 – Command bar example

As we can see, there is a command bar in the entity view of the CDS entity **account**. This can provide navigational options within an entity view, such as showing a chart, creating a new record, and deleting a record. Command bars are entity-specific. For every CDS entity, the structure of the navigation elements can be different.

Within CDS, there are three entity-specific command bars:

- **Main grid command bar:** This command bar presents all navigational options for a specific entity when an entity view with multiple records is displayed. The navigation elements can provide certain functionality to the **whole group** of displayed records.
- **Entity form command bar:** This command bar presents all navigational options for a specific entity, when an entity form with a single record is displayed.
- **Entity sub-grid command bar:** This command bar presents all navigational options for a specific entity, when a sub-grid of the entity is displayed on an entity form.

Important note

In the legacy CDS entity forms, as well as in the legacy Dynamics 365 Client for Outlook, the entity-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 CDS-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, 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 CDS, or using a third-party tool called *Ribbon Workbench*. It is certainly recommended to use the latter option, since it is much more convenient and does not lead to errors.

When adding new functionality to the command bar for a certain entity, 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.

Embedding canvas apps

Another interesting way we can extend the CDS user interface is by embedding canvas apps into CDS entity forms. This makes it possible to build tailored and visually appealing user interface elements. This customization is illustrated in the following screenshot:

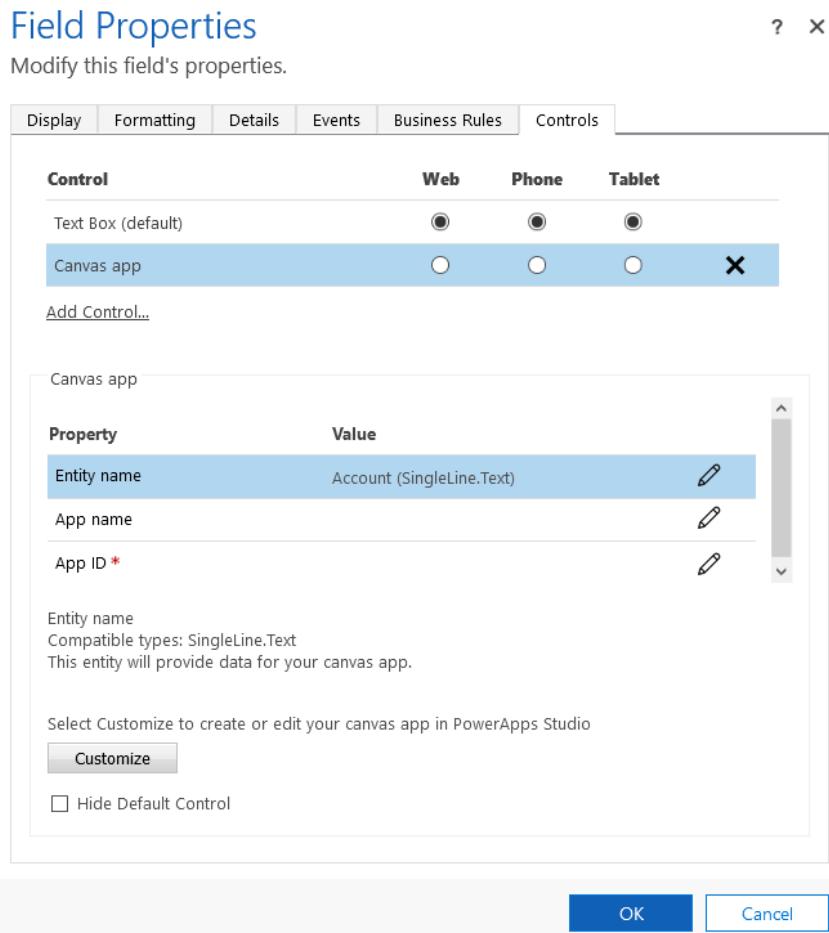


Figure 8.8 – Embedding canvas apps

Embedding a canvas app into an entity form consists of the following steps:

1. Select an entity field. It is recommended to always use a business required field and place it in a dedicated area of the entity form (for example, an empty section).
2. Edit the properties of the field to add a new custom control to the field. In this case, it must be a custom control that's of the **Canvas app** type.

3. Select the **Customize** button to start PowerApps 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 CDS field and the canvas app so that the app will be displayed on the CDS entity form instead of the CDS field.

Important note

It is not in the scope of this book to provide all the details about creating and embedding canvas apps into CDS entity 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 entity 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 entity field that's used when creating the app.

Embedding canvas apps into CDS entity forms currently has the following limitations:

- There can be only one enabled canvas app per entity form.
- The creation and embedding process must follow the high-level steps described previously. You cannot embed a canvas apps that's created outside of the described process.
- Canvas apps must be published separately. Publishing CDS customization does not publish canvas apps.
- Canvas apps must be shared with all CDS users separately. The CDS 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 CDS user interface or to implement certain client-side business logic. In the next section, we are going to focus on the server-side extensibility options within CDS.

Understanding CDS server-side extensibility

Extending a CDS solution on the server side is an option that must be used mainly in the following scenarios:

- The CDS 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 CDS solution.

In order to fulfill these advanced requirements, the CDS platform provides certain server-side extensibility options, all of which will be explained in more detail in the following sections.

CDS API interface types

The foundation for any CDS server-side extensibility is the CDS API. The CDS 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 CDS. It also allows you to call specific advanced methods going beyond the basic CRUD transactions. CDS 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 CDS transaction types.
- **Organization Service:** This is the legacy SOAP-based endpoint and is still used for certain custom development solutions (PlugIns, custom workflow actions, and so on). This endpoint provides full capabilities for all CDS transaction types.
- **Organization Data Service:** This is the legacy and already deprecated OData version 2-based endpoint.
- **Discovery Web Service:** This is a specific web service endpoint that's used to detect all CDS environments a particular user has access to. This endpoint has a Web API version, as well as a SOAP version.
- **Deployment Web Service:** This is a specific web service endpoint used for managing on-premises CDS deployments only.

- **Online Management API:** This is a specific web service endpoint used for managing cloud CDS environments.
- **Tabular Data Stream:** This is the latest endpoint, and is used for read-only access to CDS data.

Important note

The Organization Data Service endpoint still technically exists in the CDS platform, but it should not be used for any new custom development solutions.

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 **PlugIns** and **custom workflow actions** must use the organization service with the respective assemblies, since this is the only supported way to perform actions against CDS.
- Solutions that are implementing **external** communication with CDS 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, it can be the best way to use an organization service when developing using the .NET framework. For solutions that are not using the .NET framework, the only way to do this is to use the Web API. Using the discovery web service in either version can be part of the solution.
- The deployment web service (for on-premises) or the online management API (for the cloud) are used only for administration purposes.
- The **tabular data stream (TDS)** can currently only be used to **read data** from CDS, for example, within Power BI, using the **SQL Server Management Studio (SSMS)** or **SQL Server Integration Services (SSIS)**. It is, however, possible to use TDS in code to read CDS data and use it for processing.

All the solutions that implement external communication with the CDS API must contain proper authentication, as described in *Chapter 7, Microsoft Power Platform Security*. PlugIns and custom workflow actions do not need to authenticate when communicating with the CDS API since they run in the context of the CDS platform, and the CDS 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, Tools and Techniques*. The following is an overview of the most important assemblies from this package:

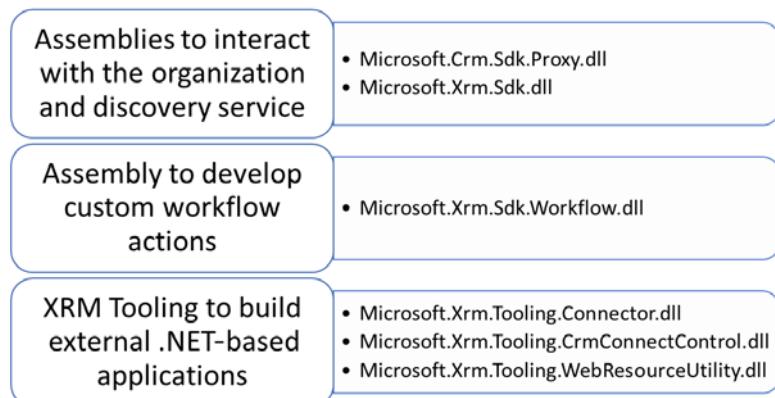


Figure 8.9 – CDS developer assemblies

There are no assemblies available for Web API development. The developer needs to use the basic .NET assemblies for *HTTP* communication, such as `System.Net.Http`.

Important note

Please refer to the product documentation for further details about the CDS API: <https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/work-with-data-cds>

In the following sections, you will learn about the various CDS server-side extensibility options.

PlugIns event handlers

PlugIns are server-side event handlers that work in a similar fashion to the CDS workflows. However, they must be developed using code. PlugIns can be used to execute transactions against the CDS 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 PlugIns consists of the following steps:

1. Develop a PlugIn event handler as a DLL assembly using Microsoft Visual Studio by following the mandatory PlugIn code structure in a .NET-based language (preferably C#).
2. Sign and compile the DLL assembly.
3. Register the PlugIn DLL assembly with CDS using the *PlugIn Registration Tool*, as described in *Chapter 4, Tools and Techniques*.
4. Register the steps of the implemented event handler for the required events using the PlugIn Registration Tool.

After performing these high-level steps, the business logic that's implemented in the PlugIn will start executing for all the registered events in the entity. The following important settings are used while registering a PlugIn step with the PlugIn 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 entity-specific. The most used generic transaction types are **Create**, **Update**, **Delete**, **Retrieve**, and **RetrieveMultiple**.
- **Primary entity:** This setting specifies the entity and which records will trigger the plugin'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 CDS.
- **Pipeline Stage:** This setting specifies in which stage during the transaction the event handler will be executed. The possible values are **PreValidation**, **PreOperation**, and **PostOperation**.
- **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 CDS workflows.

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 transaction cannot be canceled from within the event handler anymore, but 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 CDS. The business logic can implement additional transactions, such as creating, updating, or deleting records in other entities.

A PlugIn can be used for the following main scenarios:

- **Implementing advanced CDS business logic:** In this scenario, the PlugIn, which is triggered by some events in the CDS, implements certain specific business logic, such as complex calculations over other CDS records. The PlugIn communicates only with the CDS API using the organization service API from the PlugIn's execution context. Since the API is provided within the execution context of the PlugIn, no authentication is necessary. The calls of any CDS API methods are pre-authenticated.
- **Implementing outbound CDS integration:** In this scenario, the PlugIn, which is triggered by some events in the CDS, implements an outbound integration. In other words, the PlugIn sends data from the triggering CDS records to an external web service endpoint.

It is important to understand the limitations of using PlugIns with CDS, especially for calling external services:

- The PlugIn's execution is limited to a general **2-minute timeout**. After this time, every running PlugIn instance is automatically canceled.
- The PlugIn cannot include any external assemblies. There is an unsupported way to use ILMERGE to merge external assemblies with the PlugIn assembly file, but this is not recommended.

- A PlugIn 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 advanced authentication scenarios for calling the endpoints that are supported.

Due to these limitations, it might be necessary to use a different technological approach in order to implement certain complex business requirements, such as using some of the *Microsoft Azure* services instead.

In the next section, you will learn about a similar CDS extensibility option that's used to extend the capability of CDS workflows.

Custom workflow actions

Custom workflow actions are code components that are developed in a similar way to PlugIns, but they are used for a different purpose. While PlugIns are event handlers that are triggered by CDS events, custom workflow actions are not event handlers and are not triggered automatically, but rather called as actions from CDS workflows. The process of developing and registering custom workflow actions is similar to PlugIns, 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 CDS workflow. The possibilities available when using the implemented business logic are exactly the same ones that are available for PlugIns; 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 CDS workflow designer.

Azure Service Bus integration

CDS, 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 CDS platform itself.

One of the standard integrations is between CDS and **Microsoft Azure Service Bus** in order to cover the following typical scenarios:

- **Remote code execution:** In this scenario, CDS 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 CDS record being triggered, and performs any necessary processing.

- **Outgoing integration:** In this scenario, CDS 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 CDS. 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:

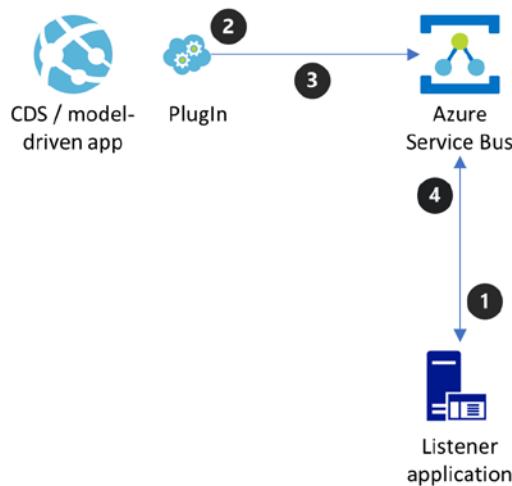


Figure 8.10 – CDS Azure Service Bus integration

As you can see, the solution consists of a CDS solution with established integration to Azure Service Bus and a listener application running on an on-premises (or cloud) infrastructure. In addition, a Microsoft Azure subscription with a configured Azure Service Bus is necessary.

The process of registering an Azure Service Bus service endpoint is illustrated in the following screenshot:

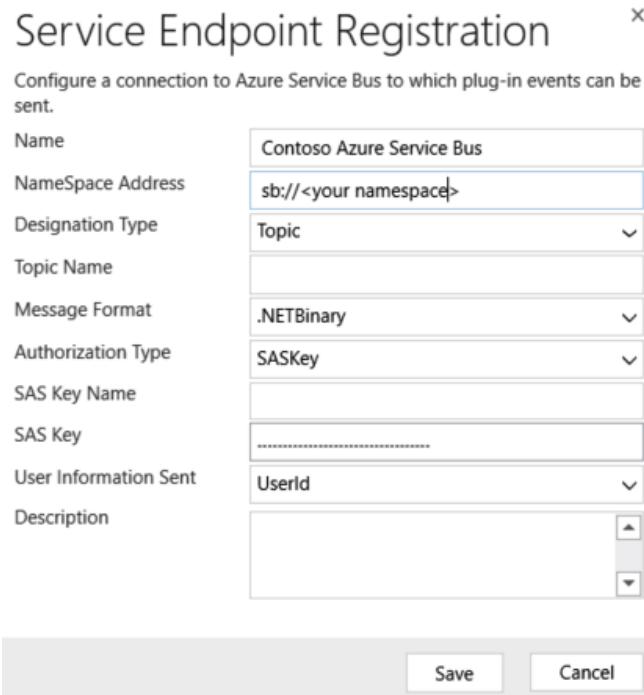


Figure 8.11 – Azure Service Bus endpoint registration

The process of establishing and operating the solution consists of the following main steps (referencing the *Figure 8.10*):

1. A listener solution must be developed and deployed on the target infrastructure.
The listener application opens a connection to Azure Service Bus and starts listening.
2. The CDS environment is configured to connect to and communicate with Azure Service Bus. This is done using the PlugIn Registration Tool in a similar way to standard PlugIns – an entity and message must be configured to trigger the communication.
3. A user performs a transaction in the CDS application that was configured for the Azure integration. A remote execution context is sent via a built-in PlugIn to Azure Service Bus.
4. The listener application receives the execution request or message and processes the content.

Important note

Due to the nature of this standard integration – that is, the use of a built-in asynchronous PlugIn – the solution always works asynchronously.

The biggest benefit and value proposition of this solution approach is that the connection between cloud and on-premises is always outbound, from the data center to 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

CDS 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 CDS side is very similar, consisting of registering with Azure Event Hub and registering the execution step to trigger the integration. When configured, the CDS 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.

Web Hook integration

CDS also offers built-in integration with a web service endpoint (**Web Hook**). This capability can, to some extent, replace the need to develop PlugIns for the purpose of outbound integration. In order to establish a Web Hook-based solution, the following steps must be performed:

1. A web service solution needs to be developed. This solution must provide the capability for `HttpHeader`, `HttpQueryString`, or `WebhookKey` authentication and must accept the CDS remote execution context as a parameter. Regarding the usual Microsoft cloud technologies, Azure Functions or Azure App Services can be used.
2. A Web Hook and the necessary steps must be registered in the CDS environment for the required entity and message type using the PlugIn Registration Tool.
3. A user performs a transaction in the CDS application that was configured for Web Hook integration.
4. The transaction triggers the integration, which forwards the remote execution context of the triggering CDS records to the web service for further processing.

In contrast to the Microsoft Azure Service Bus integration, Web Hook integration can be configured to run asynchronously but also in a true synchronous mode.

When deciding between Azure Service Bus and Web Hook, it is important to consider scalability. Since Azure Service Bus, especially when using queues or topics, is a messaging solution with large scalability, the Web Hook solution is only as scalable as the implementation of the web service permits.

Building external applications

The last CDS server-side extensibility option is to build external applications and connect to the CDS API to communicate with CDS and trigger various CDS transactions. There are several scenarios where this approach can be used:

- Build batch jobs to perform mass data processing solutions
- Build alternate desktop or web client applications communicating with CDS
- Connect existing portal solutions from third parties with CDS
- Build mobile applications communicating with CDS
- Build integration scenarios to integrate between CDS and third-party IT systems with or without an integration middleware

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

Batch jobs

Often, it is necessary to extend the standard capabilities of a CDS solution that heavily relies on real-time or near real-time, event-based extensibility. This includes solution approaches such as CDS workflows with or without custom workflow actions, CDS-triggered Power Automate flows, and plugins, all of which follow this event-based pattern.

For scenarios where event-based automation simply does not work, a batch job processing option is necessary. As an example, daily mass sending of birthday congratulation emails to contacts stored in a CDS solution cannot be automated using any of the event-based approaches. When implementing CDS batch jobs, there are different ways to use standard technologies:

- **Power Automate:** Power Automate flows must not necessarily be triggered only by CDS events, but also by using a timer trigger. The timer-triggered flow can use all the capabilities of the CDS 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 CDS using any of the preferred APIs.

There are certainly other possibilities when it comes to implementing batch jobs against the CDS 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 it's on a desktop or otherwise, should be implemented to communicate and expose some of the data and functionality of CDS. While the ideal portal extensions for CDS are **Power Apps Portals**, since they are an integrated part of the Power Platform portfolio, you may need to integrate a third-party solution or build a specific client. In this case, the CDS API can be used to cover the communication with CDS. 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, OAuth-based authentication where registration in Azure Active Directory is required is the preferred way.

Mobile clients

When developing a specific mobile solution for an internal audience, using the standard *Dynamics 365 mobile client* or creating 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 CDS Web API to perform the required communication. In order to authenticate against the API, it is necessary to 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>
- Microsoft Authentication Library 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 Apps Portals and the Unified Service Desk.

Power Apps Portals extensibility

Power Apps Portals is an extension technology that provides publicly facing portal capabilities for CDS-based solutions. Provisioning, administering, and configuring Power Apps Portals can be, to some extent, performed using administration portals and the Power Apps Portals Studio. Using this approach, standard portals or Dynamics 365-related specific portals can be configured and managed.

The Power Apps Portals technology consists of the following main components:

- **Microsoft Azure:** The runtime environment for Power Apps Portals is implemented on a shared Microsoft Azure-based technology. The customer does not have access to the individual Azure services, nor do they need individual Azure subscriptions to run the portals.
- **CDS configuration and metadata:** The content of the individual portal is fully configured within the Power Platform environment, in the CDS database. Configuration and customization can be performed by using the Power Apps Portals Studio and by making direct configuration changes in the records of the Power Apps Portals-related CDS entities.

For advanced extensibility, the Power Apps Portals technology provides the following options:

- Use frontend extensibility by implementing changes in certain Power Apps Portals-related CDS entities using the **Liquid** open source template language. Liquid can be used to customize portals in web pages, content snippets, or web templates entities.
- Use the **Portals Web API** to call the CDS Web API to trigger transactions within CDS. This capability makes it possible to extend standard Portals behavior with custom CDS-related logic.

Important note

You can learn more about the Liquid template language and about 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 Apps Portals is a complex technology and there is a need for custom development skills when it comes to custom development Portals extensibility.

Unified Service Desk extensibility

Unified Service Desk (USD) is another extension technology for CDS that offers a framework for building contact center and call center solutions, as described in *Chapter 3, Understanding Microsoft Power Platform Architecture*. USD consists of the following main components:

- **USD packages:** USD installs several solutions, along with a basic set of configuration data in the USD-specific entities. 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, UII actions, action calls, standard workflows, and UI application automations. This type of configuration is performed by configuring the settings in the USD-related entities.
- **Integrate telephony** using the UII 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 USD, such as custom application adapters (DDA adapters), custom hosted controls, and custom panel types. All these customizations require custom development efforts.

- **Integrate** USD with additional communication channels using the *Channel Integration Framework*.

Unified Service Desk 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 at: <https://docs.microsoft.com/en-us/dynamics365/unified-service-desk/extend-unified-service-desk>.

With this, we have finished analyzing the extensibility possibilities for CDS-based solutions. In the next section, we will focus on the extensibility possibilities for canvas apps and Power Automate.

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.

Introducing 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 Excel-like formulas to break the barriers for developing the apps also by non-IT professionals..
- Connections to any data sources and IT systems are implemented with the use of data connectors. There is a high number of public connectors on the market already, though if needed, custom connectors can be developed and used.

Canvas apps can be customized using the following solution approaches:

- **Use Power Apps Component Framework controls:** PCF controls, as described in the *Understanding CDS server-side extensibility* section about CDS extensibility, can be used to extend the user interface of canvas apps as well.
- **Use canvas apps custom 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. This is another type of custom component, completely unrelated to PCF components.
- **Build custom connectors:** There are more than 350 public connectors for canvas apps, Power Automate, and Azure Logic Apps. 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.

Building custom connectors

Custom connectors are wrappers around an existing or developed REST API that expose certain interface capabilities of IT systems. Custom connectors can be used to provide connection to 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. 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; that is, 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 Platform Maker Portal. You can configure the connector by creating it from scratch, importing an OpenAPI definition, importing a Postman collection, 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 environments, it can be used for canvas apps and Power Automate flows.

Important note

For further details regarding building custom connectors, please refer to the product documentation at: <https://docs.microsoft.com/en-us/connectors/custom-connectors/>.

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, the 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 a 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.

Knowing 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 CDS-based solutions.

CDS client-side interface extensibility

In this chapter, you have learned a lot about various client-side CDS 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 CDS-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 extensibility so as to avoid the most common sources of poor performance:

- Design the CDS entity forms by providing only the necessary controls; avoid overwhelming the forms with a large number of complex controls. Very large forms usually load more slowly, which obviously leads to poor acceptance.

- Use more tabs with smaller numbers 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 entity 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 IFrame 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 CDS entity forms for performance reasons – do not place canvas apps on the first entity tab. Loading canvas apps on CDS entity forms is usually slow and can distract the user.
- Use PCF controls, rather than web resources due to the nature of the PCF controls, as it loads fast in the main form load event.
- Use CDS business rules instead of JavaScript event handlers whenever possible. Business rules are an integral part of the CDS 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 CDS 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 single 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.

CDS server-side extensibility

Server-side extensibility is more complex due to so many possibilities in terms of what the CDS platform offers. In the following sections, you will learn how to select the most suitable CDS APIs, how to select the best extensibility and automation options, and what should be considered when building a performance-optimized CDS solution.

CDS API selection

When planning to build CDS server-side extensibility, there are three possibilities regarding the use of APIs and tooling:

- **PlugIns and custom workflow actions:** For these types of extensibility, 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 extensibility, 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.

- **External application, not .NET-based:** For these types of extensibility, the only option is to use the Web API, since Microsoft does not provide any support for SOAP from non-.NET solutions. In terms of using the Web API, whether you're using a .NET or non-Microsoft technology, no specific DLL assemblies are provided, so the development process needs to rely on third-party libraries.

Always select the proper authentication method against the CDS 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 CDS 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 CDS 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:

- **CDS business rules:** Simple automations based on a single entity record. They can replace JavaScript client scripting to some extent, and they also work on the server side.
- **CDS 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 flows.
- **CDS custom actions:** 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 CDS workflow, BPF, or code.
- **CDS 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 CDS workflows, CDS 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 CDS workflows. It can run as an event handler, but can also be triggered manually from the CDS application's command bar or from CDS BPF.
- **PlugIns:** 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. PlugIns 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 CDS workflows. They need to be developed using custom code in .NET. The usage scenario is the same as it is for PlugIns. The difference is that using custom workflow actions provides the flexibility needed to change the base business logic without code within the CDS workflow, while for PlugIns, every change in the business logic must be implemented by modifying certain code.
- **Azure Service Bus integration:** Can be used to implement remote automation or outbound integration scenarios in asynchronous mode only. Requires a separate Microsoft Azure license. The listener component needs to be developed using custom code in .NET.
- **Web Hook 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, for every kind of alternate client application.

In the next section, you will learn about some of the best practices in terms of performance optimization for CDS 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 CDS 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 PlugIns**.
- Implementing business logic using PlugIns registered for **Retrieve** or **RetrieveMultiple** events should be considered very carefully. Such PlugIns are known to have a serious negative performance impact. If required, performance optimization of the PlugIn'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. have made themselves 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 specific model-driven apps based on the Dynamics 365 modules 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 security role assignment to distribute the right apps to the right user groups.

Contoso Inc. have also decided to make use of the site map designer to reduce the number of entity views, forms, charts, and dashboards in the model-driven apps to further simplify the end user's experience with the apps. For the same reason, the entity forms, dashboards, and BPF will be assigned security roles to better tailor the user experience.

Contoso Inc. have made a general decision not to use the SQL Server Reporting Services technology for reporting purposes within model-driven apps. They've decided to use the views, charts, and dashboards for simple reporting, and Power BI for advanced reporting and analytics purposes.

Automations

Contoso Inc. have analyzed the various automation possibilities for CDS 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:

- CDS business rules
- CDS BPF
- Power Automate

Contoso Inc. will try to completely avoid using synchronous CDS workflows. 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 PlugIns, whenever possible. Contoso Inc. will follow all the performance-related best practices for server-side extensibility.

Client-side extensibility

Contoso Inc. have also made themselves 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 CDS business rules. Every JavaScript event handler must be carefully optimized for performance.
- If required, the use of command bar extensibility will be considered as a viable option.

Contoso Inc. decided not to use web resources and canvas apps for extending the CDS user interface.

Server-side extensibility and integrations

Contoso Inc. have understood the capabilities of the CDS 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 Apps Portals 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 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 CDS applications, implementing standard and custom automations, and using code to provide high-complexity extensibility requirements. You have seen how to extend CDS 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 of 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 CDS Web API, as one of the later extensions of the CDS API, the possibilities of developing applications integrated with CDS are now much better for non-Microsoft developers. The 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
- Perl: http://rest-examples.chilkat.io/dynamics_crm/perl/default.cshtml
- PHP: http://rest-examples.chilkat.io/dynamics_crm/phpExt/default.cshtml
- Python: http://rest-examples.chilkat.io/dynamics_crm/python/default.cshtml
- Node.js: 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 a CDS solution are. You can find full recommendations regarding this important topic in the following Microsoft product documentation article: <https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/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 customer's 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 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 enterprise 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-premise 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 experiences 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 about communicating and interacting with the customers, vendors, patients, citizens, and others. It is imperative to integrate with some electronic communication channels to support this main idea.
- 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 implicit integrations available within certain Dynamics 365 modules, as well as the possibilities and specifics of integrating Power Platform with the two other Microsoft clouds: 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 an 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 **Common Data Service (CDS)** applications, where no additional license is necessary:

- **Dynamics 365 Sales:** In this app, the whole *Sales Insights* capability is implemented using integration to 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 is implemented using Microsoft Azure services.
- **Power App Portals:** This solution consists of the metadata part, hosted within a CDS environment and the *web application*, running on a 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 they have 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 **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 CDS or Dynamics 365 and various Microsoft 365 services.

Learning about integrations with Microsoft 365

Dynamics 365 and CDS 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

CDS can be integrated with both **Exchange Online**, which must be in the same tenant as CDS, or with **Exchange on-premise**. A limited integration (emails only) can be established to 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:

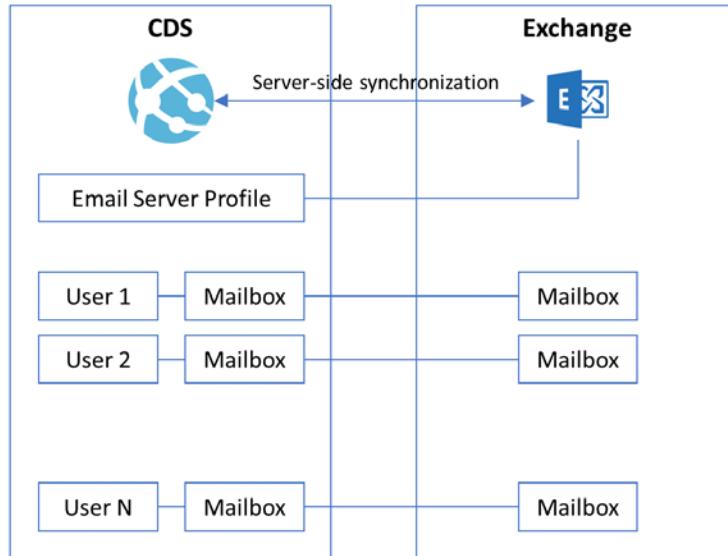


Figure 9.1 - CDS integration with Exchange

As we can see, on the CDS side, there is an entity called **Email Server Profile** that specifies the connection to an **Exchange** (or other) server. It is very well possible to have a heterogenous configuration with multiple email server profiles of different types (for example, an Exchange on-premise 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 automatically has a corresponding **Mailbox** record created and assigned, which represents the connection between the CDS user and the corresponding email account in **Exchange**. The process of establishing this integration 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-premise 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 CDS application users can benefit from synchronizing important data such as emails, appointments, contacts, or tasks between those two systems.

Important note

It is important to understand that every Exchange mailbox can only be integrated with one CDS environment. The *Enable and Test* feature in CDS makes it possible to reconfigure a mailbox so that it's detached from the previous CDS environment and attached to the new one.

Now, let's have a look in the following section at the integration to **SharePoint**.

Integrating with SharePoint

Dynamics 365 applications can be integrated with **SharePoint** for 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 Dynamics 365 environment. A hybrid integration with SharePoint on-premise is also available.

The process of establishing integration consists of the following steps:

1. Create a dedicated SharePoint site.
2. Perform the configuration setting in Dynamics 365 using the dedicated SharePoint site. Part of the configuration is to decide for which Dynamics 365 entities the integrated document management will be enabled.

After this simple configuration, integrated document management will be available from the main entity forms for records of each enabled entity.

Now, let's have a look at another integrated document management feature, this time for storing private documents.

Integrating with OneDrive for Business

Another solution that provides integrated document management for Dynamics 365 applications is integration with **OneDrive for Business**. The biggest difference between SharePoint and OneDrive for Business is that while SharePoint is a collaborative platform where normally every user has access, OneDrive for Business is a private document storage where every user has their own protected storage area.

In order to use the OneDrive for Business integration, the SharePoint integration must be enabled. The OneDrive for Business integration can be easily configured in a few simple steps. For the end user, both the SharePoint and the OneDrive for Business 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 for Business in one single list, as shown in the following screenshot:

The screenshot shows a Dynamics 365 Opportunity record for '4G Enabled Tablets'. The top navigation bar includes 'Opportunity Sales Process' (Active for 9 months), 'Qualify', 'Develop', and 'Propose (9 Mo)'. The 'Opportunity' tab is selected. The 'Documents' tab is underlined, indicating it is active. Below the tabs, there are buttons for 'Show Chart', 'Document Location', 'Open Location', 'Add Location', 'Edit Location', and 'Refresh'. A 'Document Associated Grid' section displays three documents:

Name	Modified	Modified by	Path	Source
Confidential Opportunity Background.pptx	31.07.2020 15:33	Robert Rybaric	personal/admin_robertrybaric_o...	OneDrive
Opportunity Calculation.xlsx	31.07.2020 15:32	Robert Rybaric	sites/Dynamics365/opportunity...	SharePoint
Opportunity Summary.docx	31.07.2020 15:32	Robert Rybaric	sites/Dynamics365/opportunity...	SharePoint

Figure 9.2 - Integrated SharePoint and OneDrive for Business 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 for Business.

Integrating with Microsoft Teams

Another important integration provided for Dynamics 365 applications is integration with **Microsoft Teams**. There is a possibility to configure a basic or enhanced collaboration experience. This integration offers the following capabilities:

- Open a Teams channel and assign a Dynamics 365 record or entity view to it. This will add the entity main form or the selected entity view to the channel. The members in the team can work around that record or view using chat, document management, and other options. The documents that are created in such a channel that are connected to Dynamics 365 will be also available from the entity form outside of the Microsoft Teams environment. It is required to have SharePoint integration enabled to have access to those documents.

- The enhanced experience makes it possible to work from within Dynamics 365 to establish the functionality described in the previous section.
- Add a Dynamics 365 **dashboard** to the Teams environment.

This collaboration capability is illustrated in the following screenshot:

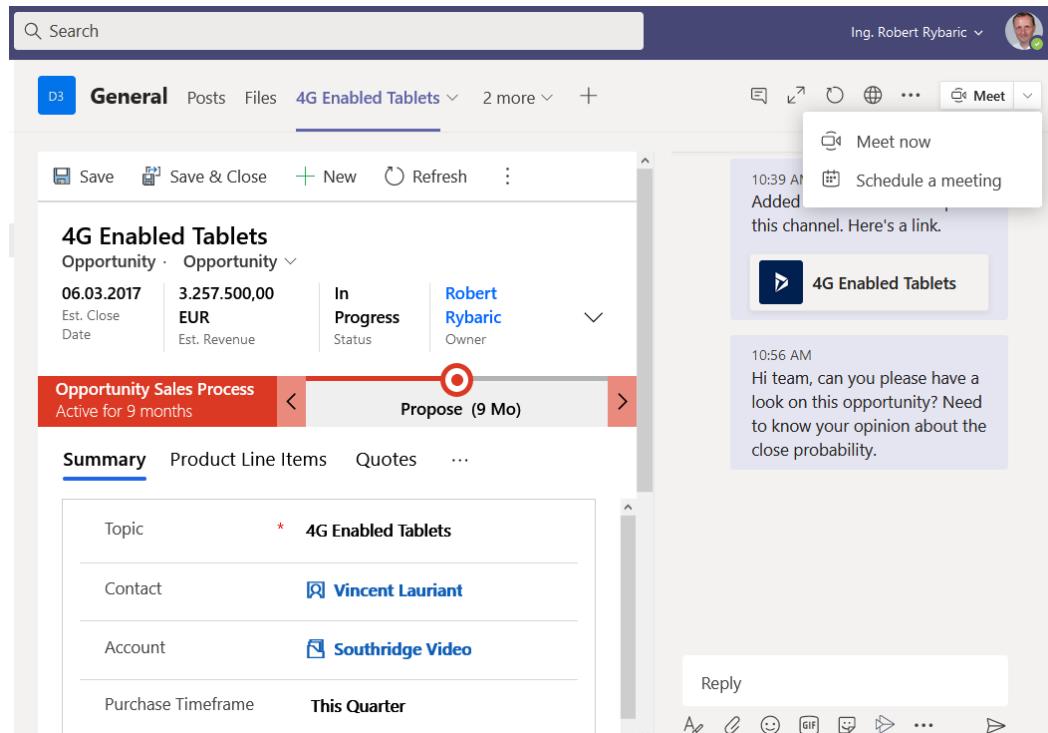


Figure 9.3 - Dynamics 365 integration with Microsoft Teams

As you can see, an integration was created, with an opportunity that was enabled in Microsoft Teams. The opportunity entity main form is included in the Teams channel. All the usual capabilities of Microsoft Teams, such as the chat capability, are available within this channel.

The integration configuration consists of the following simple steps:

1. Enable the basic or enhanced experience in the Dynamics 365 system settings.
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.

Integrating with Microsoft OneNote

Microsoft OneNote is a note-taking application. OneNote can be integrated with Dynamics 365 to provide all the benefits of OneNote from the context of a Dynamics 365 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. The OneNote integration requires that you have the SharePoint integration enabled.

After enabling this integration, the capability is available from within the **Timeline** control on the entity form, as illustrated in the following screenshot:

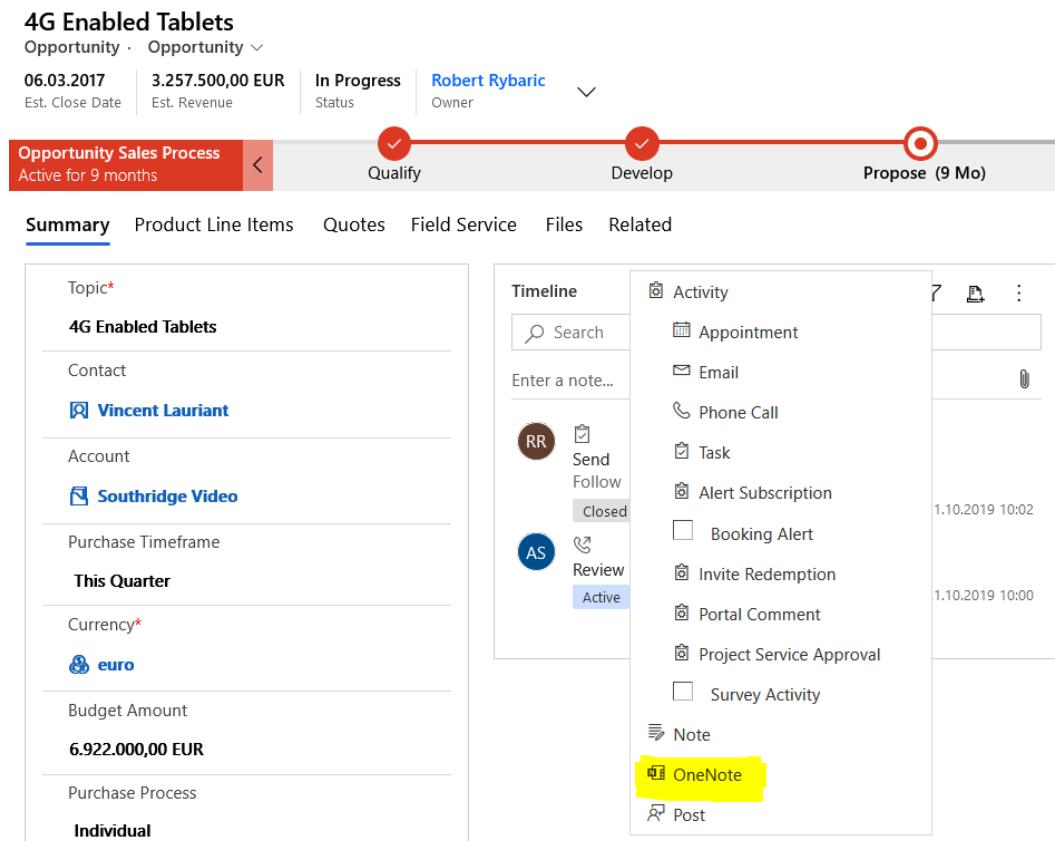


Figure 9.4 - Dynamics 365 integration with Microsoft OneNote

As we can see, we can open the OneNote file that was created for a particular record and use it to take notes, as well as use it with all the previously mentioned OneNote capabilities. The OneNote file is stored in the SharePoint folder that's attached to the CDS record.

The configuration steps for the OneNote integration process are as follows:

1. Turn on the OneNote integration in the document managements settings in Dynamics 365.
2. Select the entities to be enabled for the integration.

After following these simple configuration steps, integration will be enabled, and the capability will be available from the entity records of the enabled entities.

Integrating with Skype/Skype for Business

Skype (consumer) or **Skype for Business** can be integrated with Dynamics 365 applications to provide a click-to-call capability for users. This capability makes it possible to trigger a phone call from within any Dynamics 365 record, where telephone numbers are stored. After clicking on the phone icon to the right of the telephone number field, the configured application (Skype or Skype for Business) will be ready to open the phone call. At the same time, a quick create form for the Phone Call entity 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:

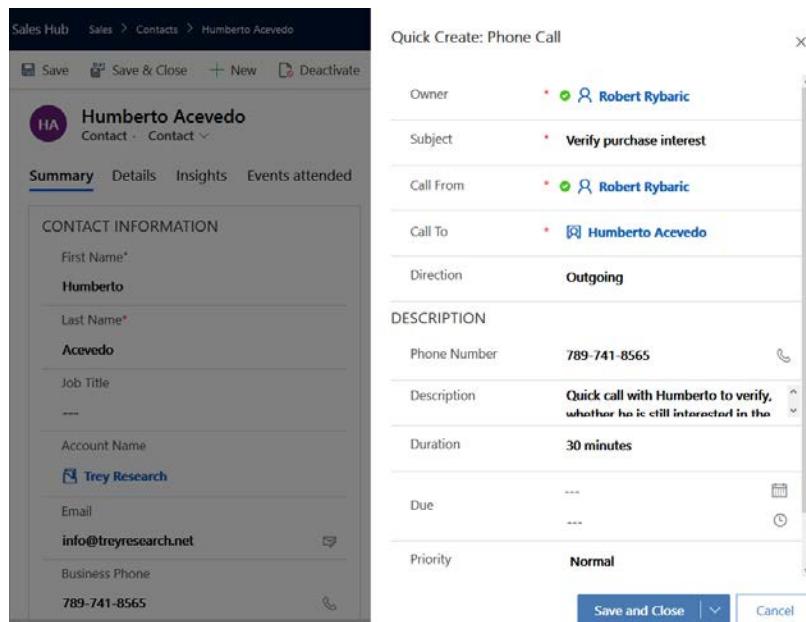


Figure 9.5 - Dynamics 365 integration with Skype/Skype for Business

As you can see, by clicking on the Business Phone field'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** fields) 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 **Dynamics 365 system settings**.

Integrating with Microsoft Yammer

Microsoft Yammer is considered an enterprise social network, giving the possibility to employees to have chat-based conversations, share information, and work on documents. The capabilities of Yammer overlap slightly with Microsoft Teams. After enabling integration from Dynamics 365 to Yammer, you can engage in Yammer conversations and collaborate from within a Dynamics 365 record, as shown in the following screenshot:

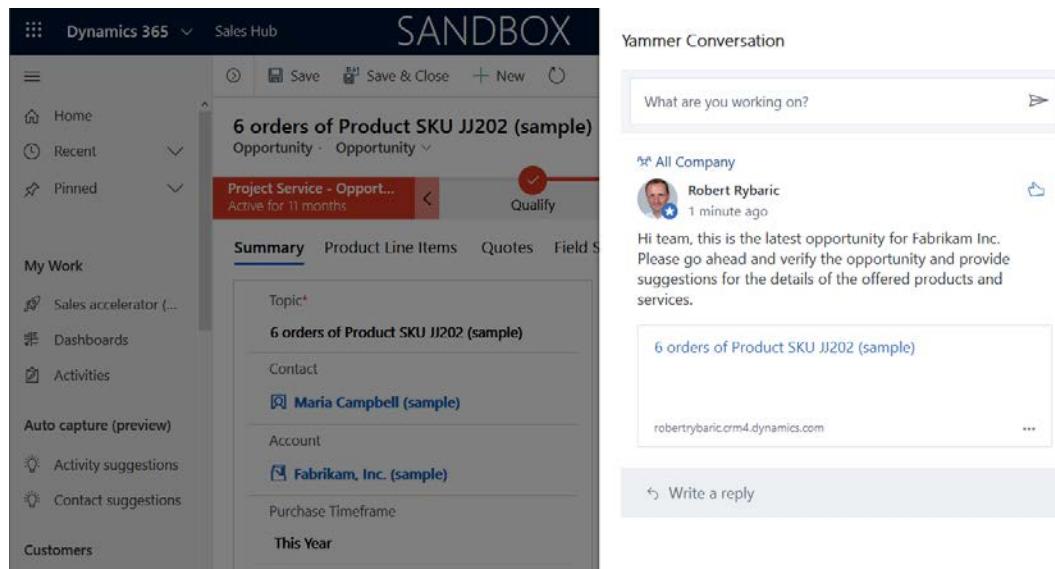


Figure 9.6 - Dynamics 365 integration with Yammer

As we can see, a Yammer conversation and collaboration was started from an opportunity, which is an entity that's enabled for the Yammer integration. The conversation, together with a link to the Dynamics 365 record, will be available also from within the Yammer application itself.

The steps to enable this integration are as follows:

1. Enable a connection to Yammer in the Dynamics 365 system settings.
2. Configure selected entities to be enabled for Yammer.
3. Publish all customizations.

After completing these configuration steps, the Yammer conversation will be enabled from within the Dynamics 365 records for all enabled entities.

Integrating with Microsoft 365 Groups

Microsoft 365 Groups is a feature of the Microsoft 365 cloud. Groups can be used for many different purposes. You have already learned how to use groups to manage the provisioning of the user records in various CDS environments within a tenant. Besides those technical purposes, you can also use Microsoft 365 Groups for group collaboration, and sharing files, using a group calendar and email address. All of those features can be leveraged from the context of Dynamics 365 records, using an integration between Dynamics 365 and Microsoft 365 Groups. The integration requires that we have an existing integration with Exchange to use the group email and calendar, and also with SharePoint for storing group documents.

The configuration consists of the following steps:

1. Install the *Office 365 Groups* solution in a CDS environment. This can be done from within the Power Platform Administration Center.
2. Configure the integration settings in the Dynamics 365 settings area to specify which entities the integration will be established for. For every entity, you can specify whether a new group should be created automatically for each new record in the entity.

After finishing the integration, you can work in the Microsoft 365 Groups space from within Dynamics 365 records, but also from within the Web Outlook and Calendar apps, as well as from Outlook for desktop or mobile devices.

In the next section, we will pay attention to integration possibilities between CDS applications and various Microsoft Azure cloud services.

Learning about integrations with Microsoft Azure

CDS 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.

Important note

All the described integration capabilities require the use of CDS and Microsoft Azure in the same tenant.

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 from CDS 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 CDS applications. For details, please refer to *Chapter 8, Microsoft Power Platform Extensibility*.
- **Web Hooks:** 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, very important integration possibilities, leveraging standard integrations.

Integrating with Azure Blob storage

CDS 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 CDS applications distinguishes between relational data and files, in case the attachments are heavily used, it might occupy the available storage in the CDS database far too quickly. For such situations, you can completely offload all attachments from the business records to a dedicated **Azure Blob storage**. Since Azure Blob storage is much cheaper than CDS storage (or CDS 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 Blob storage.
2. Install the **Attachment Management** solution provided by Microsoft Labs free of charge. This solution is available from Microsoft AppSource from the following URL: https://appsource.microsoft.com/en/product/dynamics-365/microsoft_labs.96257e65-dbbe-43db-b775-77cf1609530c.
3. Configure the solution from within the CDS application.

Important note

Offloading the attachments into Azure Blob storage will remove the offloaded content from the overall global search capability of the CDS platform.

After installing and configuring the solution, all the file attachments will be automatically stored in the Azure Blob storage.

Integrating with Azure SQL

One of the limitations of CDS applications is their inability to directly work with the underlying relational 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 and data warehousing scenarios. There is, however, the possibility to have a live replica of the CDS database in your own **Azure SQL** database with full access. This can be achieved using a solution that consists of the following components:

- A Dynamics 365 or CDS application
- An Azure SQL database or Microsoft SQL Server on a VM database and an Azure Key Vault instance
- A CDS solution from Microsoft AppSource

As you can see, the customer needs an own Microsoft Azure subscription to establish and run this solution.

Important notes

Only entities enabled for change tracking can be part of the replication process.

Only the business data, without the security context, will be exported to the external Azure SQL database.

The process of establishing this capability consists of the following steps:

1. Use your Azure subscription to provision Azure SQL or Microsoft SQL Server on a VM database instance. In addition to this, provision an Azure Key Vault instance.
2. Install the **Microsoft Dynamics 365 - Data Export Service** solution provided by Microsoft Labs free of charge. The solution is available from Microsoft AppSource from the following URL: <https://appsource.microsoft.com/en-US/product/dynamics-365/mscrm.44f192ec-e387-436c-886c-879923d8a448>.
3. Create an export profile within the CDS application. In the profile, entities and their relationships can be specified for synchronization.

After installing and configuring the solution, the replication process will start working. This replication process runs in a near-real-time fashion 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.

Integrating with Azure Data Lake

Another way to automatically export CDS data into external storage is by using the built-in capability to export to **Azure Data Lake**. Compared to the other Azure integration solutions, this export capability can be configured directly in the Power Platform *Maker Portal*, without the need to install any CDS solutions. This capability can export data from CDS entities to Azure Data Lake, where the data is stored as CSV files in CDM format.

Important note

Only entities enabled for change tracking can be part of the replication process.

Only the business data, without the security context, will be exported to the external Azure Date Lake repository.

This capability can be established by using the following steps:

1. Create a Microsoft Azure storage account of the **Storage V2 (general purpose)** type. The **Hierarchical namespace** feature must be enabled.
2. Create an export configuration in the *Maker Portal*, specifying all required entities.

After configuring the export, an initial synchronization will be performed, followed by continuous replication. The exported data will be available in the Azure storage in a collection of folders, where the data file for each entity will be stored in an entity-specific folder. In addition, in each entity 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 CDS data, which is not constantly updated. The content of the Azure Data Lake can be viewed using the *Azure Portal* or the *Microsoft Azure Storage Explorer*. The data can be used for analytics and reporting purposes.

Integrating with Azure Cosmos DB

CDS and Dynamics 365 applications are considered operational solutions, where all the data is just created, regardless of the number of existing records in the database. In order to get rid of old data, you can manually delete records or use the bulk delete capability. There is, however, no standard capability to archive old data in another repository and then remove that data from the operational CDS database. In order to address this gap, you can archive records from CDS entities using a solution consisting of the following components:

- A Dynamics 365 or CDS application
- Your own **Azure Cosmos DB**, Blob storage, key vault, and API apps
- A CDS solution from Microsoft AppSource

For this solution, again, you need to have your own Microsoft Azure subscription.

This solution can be established using the following steps:

1. Register the archiving solution in the Azure Active Directory to obtain the *application ID* and *client secret*. Create an application user in the CDS using these credentials.
2. Install the **Dynamics 365 Data Archival and Retention** solution provided by Microsoft Labs free of charge. The solution is available from Microsoft AppSource from the following URL: https://appsource.microsoft.com/en-us/product/dynamics-365/microsoft_labs.dataarchival.

3. Using an Azure resource template, as provided in the solution documentation, create all the necessary *Azure resources*.
4. Configure the archiving parameters in the Azure Cosmos DB settings.

After all the components have been installed and configured, the solution will start automatically archiving data from CDS into Cosmos DB and removing that data from the CDS database.

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 that 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">• Does not need an Azure license• Cannot easily integrate with Microsoft Azure services• Native integration with canvas apps• Provides options for human interactions• Can be deployed with CDS solutions• Approval process• Provides an app for mobile devices	<ul style="list-style-type: none">• Requires an Azure license• Can be easily integrated with other Azure services to build complex solutions• Enterprise scalability• Can be developed using code with Visual Studio

Figure 9.7 - 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 Platform or Office 365. Azure Logic Apps is more suitable for advanced integration scenarios, including additional Azure components, along with Logic Apps and with better support for pro IT developers.

The support for CDS applications is given by the CDS data connector, which works the same way as it does for Power Automate.

Important note

The *Common Data Service (current environment)* connector is not available for Logic Apps, only for Power Automate or canvas apps, when using it from within a CDS solution.

Integrating with Azure API Management

Azure API Management is a robust Azure service for building externally facing APIs to provide integration endpoints to internal cloud solutions for your own on-premise 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, CDS solutions. The benefits of doing this are as follows:

- API Management can be used to expose a protected subset of the CDS 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 CDS solution to expose some of the CDS APIs can be established using the following steps:

1. Create an API Management instance in the own Azure subscription.
2. Register the integration application in the Azure Active Directory to obtain the *application ID* and *client secret*.
3. Create a new application user in CDS 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 to 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 the CDS. 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 CDS 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 CDS applications in the frontend side, as well as on the backend, with third-party IT on-premise or cloud systems and solutions.

Presenting frontend integration patterns and solution approaches

In this section, we are going to discuss the most common frontend integration patterns for CDS applications. Frontend integration can be used for a variety of reasons, such as to embed some third-party content on a CDS 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 CDS

CDS applications allow us to embed third-party content into CDS entity forms and dashboards. In both cases, the integration is implemented using the *IFrame* approach. An **IFrame** is an area on the entity form or dashboard where *URL-based* content can be embedded into. In the simplest case, this content is static and does not depend on the context of the entity record, nor any other context. Using static content might be useful for embedding into dashboards, but for entity forms, the requirement would mainly be to make the content somehow related to the entity type and the currently open 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 CDS record, the entity type, the internal name of the CDS environment, and the language code of the CDS 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:** In case 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 entity form and configure the IFrame's URL properly.

An example of the standard IFrame configuration is documented in the following screenshot:

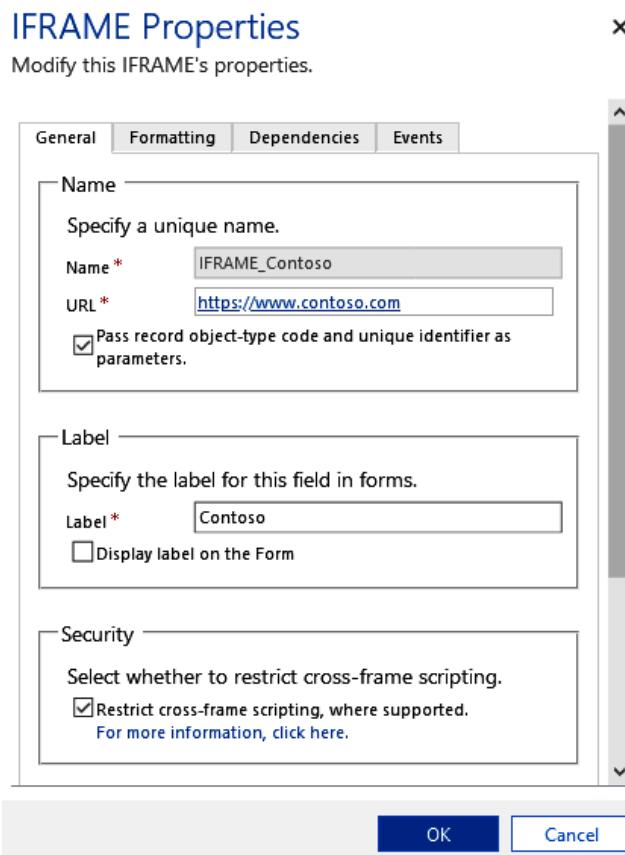


Figure 9.8 - IFrame configuration for passing the standard context

As you can see, the configured URL is `https://www.contoso.com` and the **Pass record object-type code and unique identifier and 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.contoso.com?id=90f1fa99-3788-e911-a818-000d3ab2dff  
3&typename=account&type=1&orgname=org6d9723d9&OrgLCID=1033&UserLCID=1033
```

As we can see, the *record ID*, *entity type name* and *type code*, the internal CDS *environment name*, as well as the *language codes* 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 plugins, 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 CDS content into third-party containers

It is sometimes required to embed parts of a CDS 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 CDS application into a central place that's visible for every user. These requirements can be addressed using the URL-addressable CDS resources. CDS allows us to address the following application elements with a specific URL:

- **Entity forms:** It is possible to embed an empty entity form, or a form containing data, from a selected record. If the entity 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. Entity forms can also be embedded on mobile devices – the URLs are formatted differently.
- **Entity views:** It is possible to embed system or personal views. For views, it is also possible to specify the visibility of the navigation elements. Entity views can also be embedded on mobile devices using different URL formatting.

- **Reports:** It is also possible to embed any of the SQL Server Reporting Services reports that are available in the CDS solution. In the URL, you can specify whether the report will be executed directly or whether they will be first offered the possibility to configure the filtering for the report.
- **Dashboards:** Dashboards can only be embedded on mobile devices using mobile-specific URL formatting.

It is important to understand that embedding CDS content does not automatically provide access to that content to everybody, but just for users with the proper Power Apps or Dynamics 365 license, which are enabled as users in the respective CDS environment.

Event-driven and on-demand frontend integration

You learned about the forms scripting and command bar extensibility capabilities of CDS solutions in *Chapter 8, Microsoft Power Platform Extensibility*. JavaScript event handlers can not just manipulate the content of CDS entity forms, but also implement communication with web service endpoints. They can not only call the CDS client-side Web API, but also any external web service. There are several popular methods for calling external web services from a JavaScript code, including the following:

- **Ajax:** Ajax was used traditionally to make server-side calls from CDS 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 make 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 an entity form to an external web service endpoint for processing.
- Retrieve data from an external web service endpoint and fill in some entity form controls with the data.
- Retrieve some data from the CDS application using the *client-side Web API*, combine this data with some data from the entity form, and send it all to an external web service endpoint.

In the next section, you will learn about the new Power Automate solution component known as UI flows, which is a service used to implement **robotic process automation (RPA)**, as well as about the **Unified Service Desk (USD)** client, which can help you implement frontend integration.

Using UI flows and Unified Service Desk

UI flows is an extension of the Power Automate cloud service. While the standard Power Automate works with the concept of data connectors, which expect the integrated systems to have an API, UI flows are intended to integrate legacy IT systems that have no usable API.

A UI flows solution is able to perform frontend automation of desktop and web applications. The solution requires a local installation of some components on the desktop PC where the legacy application will be automated. For remote automation scenarios where you're targeting remote computers, the *On-Premise Data Gateway* needs to be installed.

The solution can be used to automate processes with Power Automate between API-based IT systems using *data connectors* and non-API-based IT systems using frontend automations. UI flows configuration consists of recording all the necessary clicks and data entries or retrieves on the legacy IT system. Once the UI flows setup is working, the overall automation will be implemented in a Power Automate flow, calling UI flows as part of the overall automation.

Important note

For more details about Power Automate UI flows, refer to the product documentation: <https://docs.microsoft.com/en-us/power-automate/ui-flows/overview>.

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, and **Windows Presentation Foundation (WPF)**)
- Java applications
- Web applications (including CDS itself, Silverlight, and Java applets)
- Citrix-hosted applications

The capabilities are similar to those of UI flows. 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 used client-side integration patterns and technical possibilities. The next topics we will investigate in this chapter is the backend integration of CDS or Dynamics 365 applications with other IT systems.

Presenting backend integration patterns and solution approaches

In the following sections, you will learn about the most typical integration patterns used within CDS-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.

Remote procedure call pattern

The **remote procedure call (RPC)** pattern is the simplest integration pattern, where one application is directly calling 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:

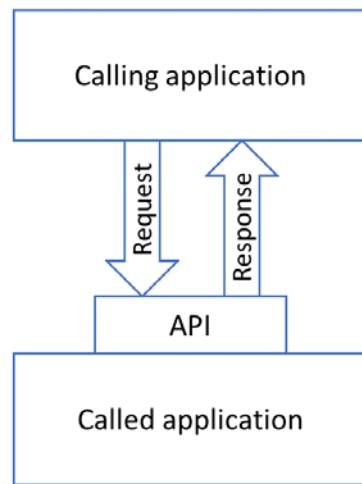


Figure 9.9 - Remote procedure call pattern

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 CDS solutions, there are two basic outbound implementation approaches, as shown in the following diagram:

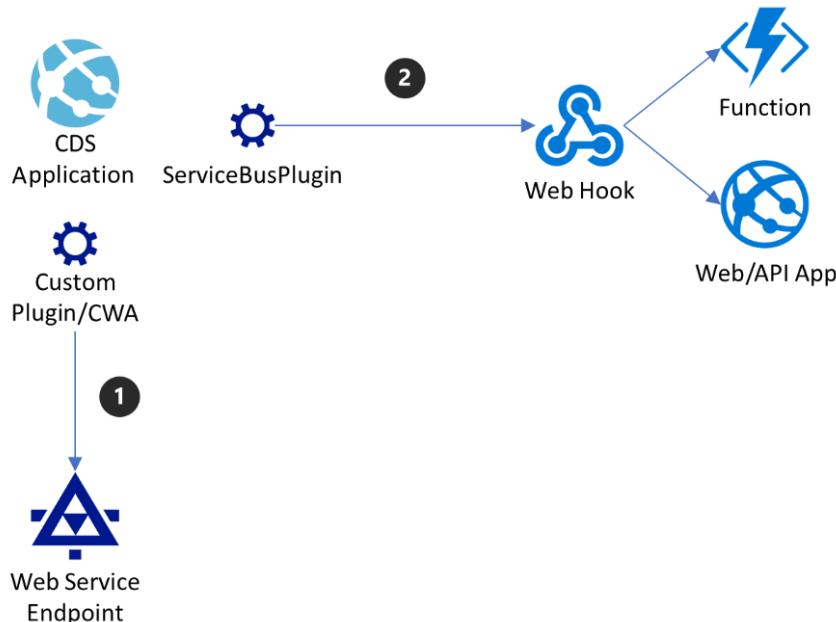


Figure 9.10 - Outbound RPC solution approaches

As we can see, the following suitable implementation options are available:

1. **Custom PlugIn or custom workflow action (CWA):** For this solution, it is required to develop a custom CDS PlugIn or custom workflow action. The PlugIn 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 type HTTP or HTTPS with a DNS URL (no IP address) and must be implemented in the cloud, or for on-premises deployment, must be published to Internet. Azure API Management can be used to support building the on-premise 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. **Web Hook integration:** For this solution, the built-in plugin for Azure integration can be used, but for integration with a Web Hook. 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 Web Hook endpoint include Azure Functions, Azure Web Apps, and API Apps.

The inbound implementation of this pattern is very simple, as shown in the following diagram:

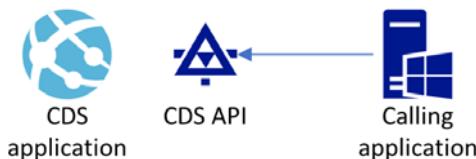


Figure 9.11 - Inbound RPC solution approach

As we can see, calling the native CDS API is basically an *inbound RPC solution* approach since the vast majority of the CDS 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 Microsoft Power Platform Architecture*, need to be considered.

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 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:

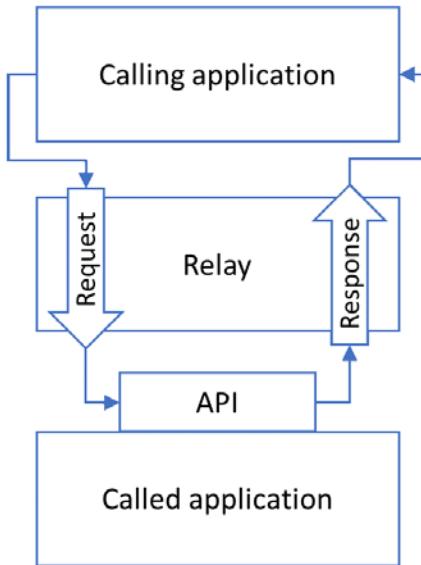


Figure 9.12 - Relay pattern

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 CDS solutions, this pattern can be implemented using *Azure Service Bus integration*, as shown in the following diagram:

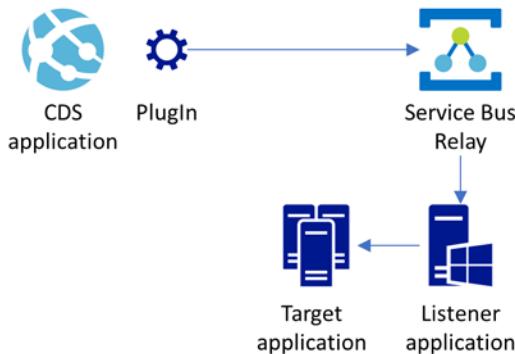


Figure 9.13 – Outbound relay solution approach

As we can see, the solution consists of the CDS 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 that we open inbound communication 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 CDS PlugIn.

There are three different ways to implement the PlugIn:

- **Built-In PlugIn (ServiceBusPlugIn):** This solution does not require any custom development on the plugin 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 PlugIn:** This solution requires developing an Azure Aware PlugIn, which is a specific PlugIn type that reuses the standard Azure integration capabilities. This solution makes it possible to implement also a synchronous communication pattern and to process the return values coming from the listener.
- **Custom PlugIn:** This is the most flexible solution as it makes it possible to implement synchronous communication and any required business logic in the PlugIn. The solution requires that we implement all the logic and the connection to the Azure Service Bus in the PlugIn 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:

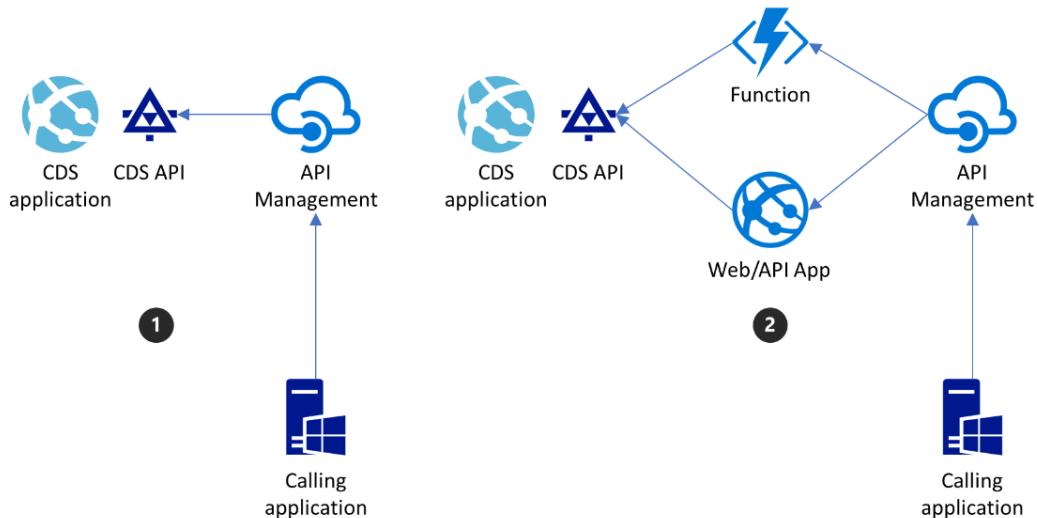


Figure 9.14 - Inbound relay solution approach

As we can see, there are the following options:

- **API Management only:** For this solution, API Management is used to isolate the CDS API from the external integrated applications. The role of this component is to publish certain selected CDS 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- or 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 components, such as Azure Logic Apps, but rather *components running synchronously*.

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 in the communication. The pattern uses messaging middleware, as shown in the following diagram:

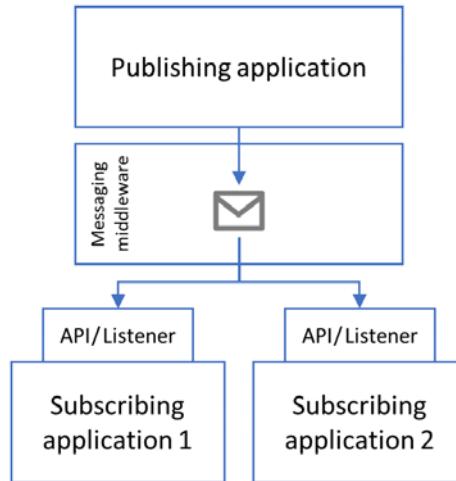


Figure 9.15 - Publish-subscribe pattern

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 Hub, as illustrated in the following diagram:

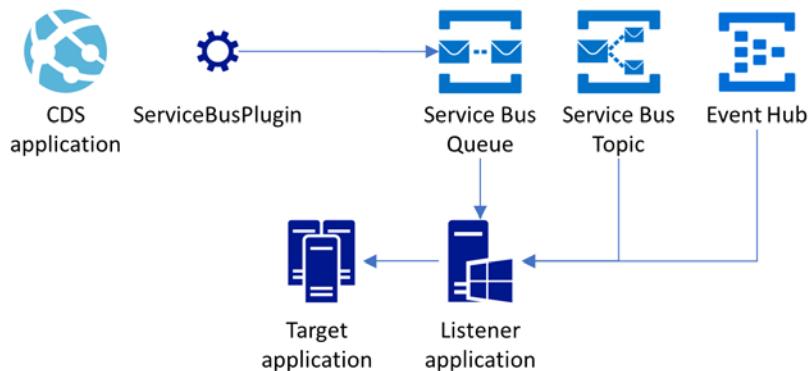


Figure 9.16 - Outbound publish-subscribe solution approach 1

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 Hub**. In all three variations, the integration on the CDS 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, but the messages from CDS will wait until they are retrieved. The listener will receive the messages in the native CDS 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:

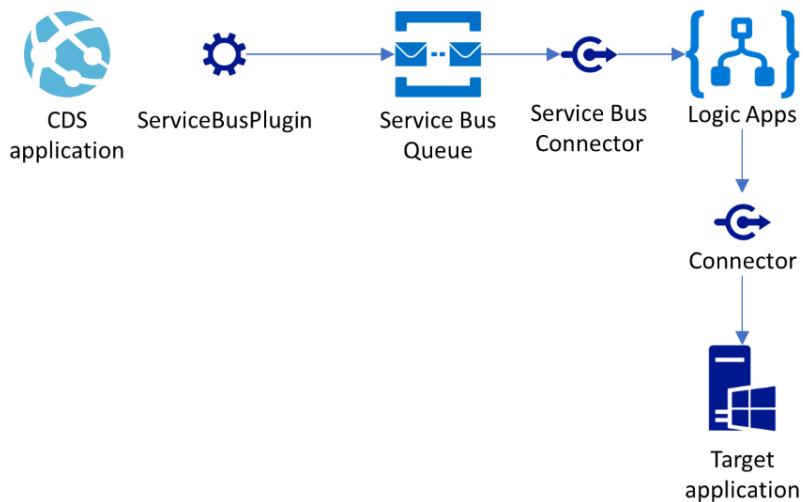


Figure 9.17 - Outbound publish-subscribe solution approach 2

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 350+ public connectors or a custom connector, if the target application does not provide any 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 the Azure Service Bus and only use *Azure Logic Apps* with a CDS connector, as illustrated in the following diagram:

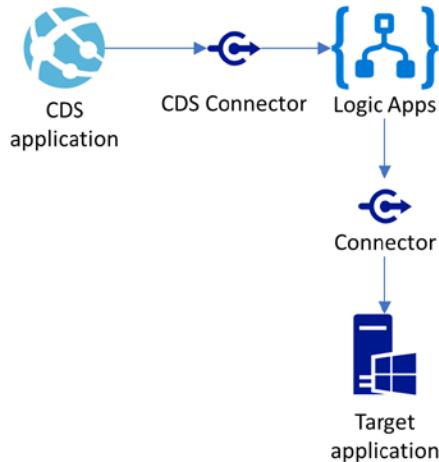


Figure 9.18 - Outbound publish-subscribe solution approach 3

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 CDS 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:

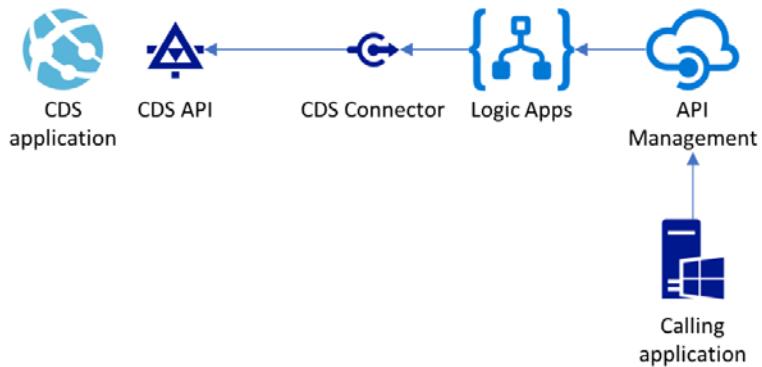


Figure 9.19 - Inbound publish-subscribe solution approach

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 secured.

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:

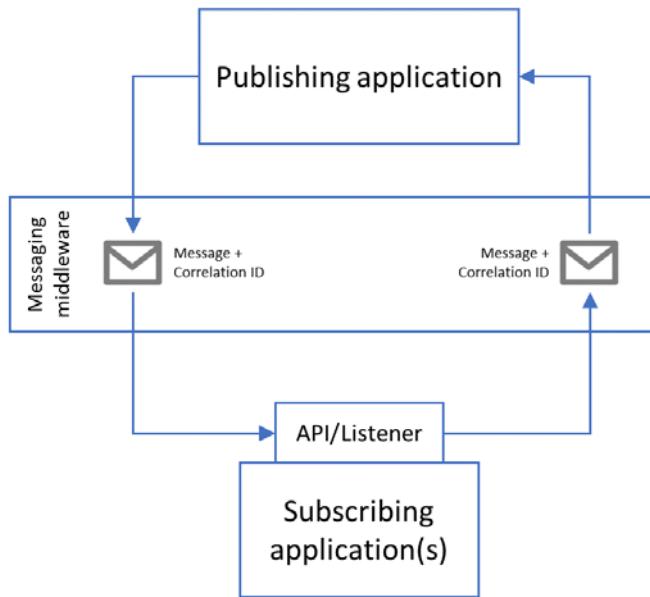


Figure 9.20 - Request-callback pattern

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 patterns require 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 in 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 CDS applications, the pattern for outbound, as well as for inbound, can be implemented as a combination of the solution approaches from the previous patterns. The following diagram shows the possible implementations:

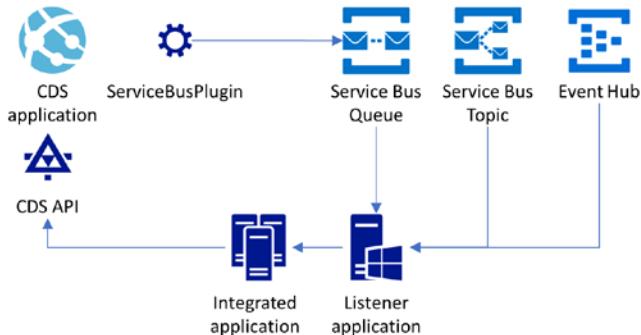


Figure 9.21 - Outbound request-callback solution approach

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 CDS API. A variation of this solution approach would be to put an API Management component in front of the CDS 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:

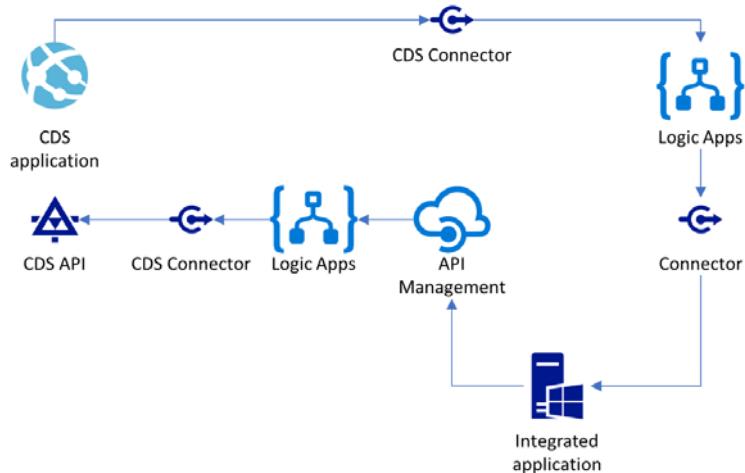


Figure 9.22 - Inbound request-callback solution approach

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 is synchronized at once using a *scheduling* approach. This pattern is illustrated in the following diagram:

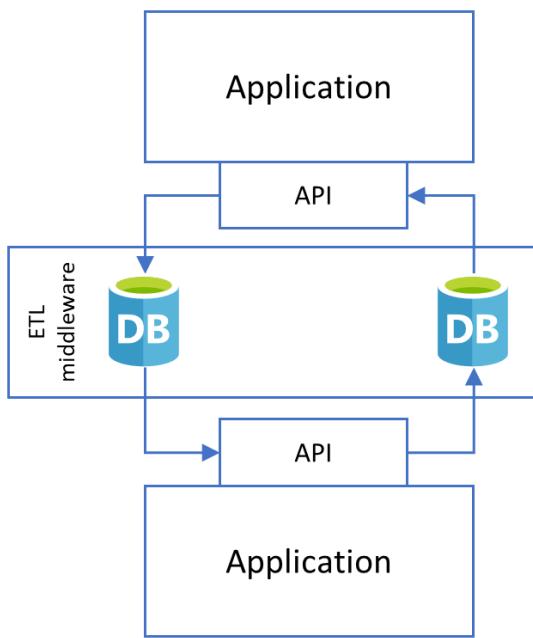


Figure 9.23 - Data integration pattern

As we can see, the main component of this solution is an **extract-transformation-load (ETL)** middleware, which is actively managing the integration processes. There are several integration solution approaches for CDS applications that follow this pattern. You will learn about some of those approaches in the subsequent sections.

Virtual entities

Using **virtual entities** is a native CDS capability as it allows you to include data from an external data source natively in the CDS application. This capability is illustrated in the following diagram:

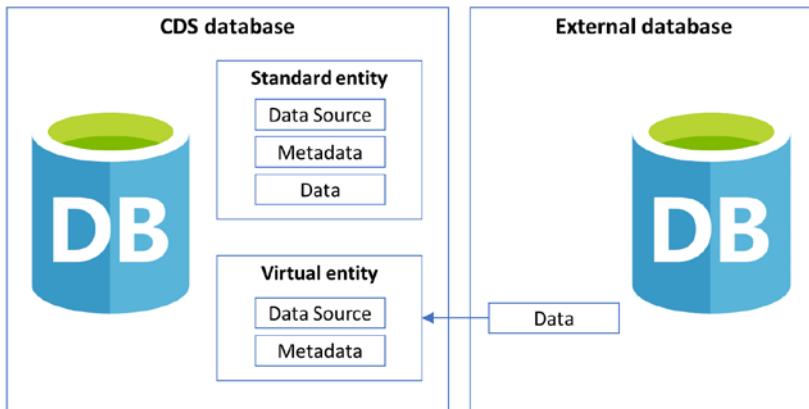


Figure 9.24 - Virtual entities

As we can see, a standard entity has the whole configuration, metadata, and the data itself stored in the CDS database directly. A virtual entity has the definition and metadata stored in CDS, but the business data resides in an external database. The purpose of this capability is to bring external data directly into the CDS application and make it part of the standard user interface, completely transparently for the end user. Virtual entities have several restrictions that must be considered when it comes to deciding whether to use this feature:

- The data from virtual entities is read-only in the CDS application.
- The number of supported data types is limited. Certain CDS-specific data types, such as *Currency*, *Image*, and *Customer*, are not supported.
- Several entity features are not supported, such as auditing, the activity type of the entity, change tracking, offline capability, field security, and more.
- Virtual entities can only be organization-owned, not user/team-owned.

Virtual entities use the concept of **Data Provider Plugins**, which are used to make the connection between CDS and the external data source. Currently, the following plugins are available:

- **A standard OData 4.0 plugin:** This plugin is available by default from the CDS platform. This plugin makes it possible to integrate with any external data source that supports an OData 4.0 web service endpoint.

- **Azure Cosmos DB plugin:** This plugin is dedicated to connecting to the Azure Cosmos DB cloud databases. By default, the plugin is not installed in the CDS platform, so a separate installation from AppSource is necessary: https://appsource.microsoft.com/en-us/product/dynamics-365/mscrm.documentdb_data_provider.
- **Custom PlugIn:** If neither of the existing Data Provider Plugins are suitable to connect to a specific data source, you can develop a custom plugin. 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 plugin.

Virtual entities can be customized in a similar way as standard entities, using the usual CDS customization tools.

Important note

At the time of writing this book, the virtual entities concept has been extended to enable the creation of virtual entities for Dynamics 365 Finance and Supply Chain Management, as well as for the Dynamics 365 Business Central entities. It also covers the create, update, and delete transaction types. This is a very important new feature that enables completely new and simplified integration scenarios between the various Dynamics 365 applications.

Using *virtual entities* is a very interesting approach since it has the benefit that the external data does not need to be physically replicated in CDS, but can reside in the original repository. Should the limitations of virtual entities 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 CDS data into cloud services, such as Azure SQL, Microsoft SQL Server on Azure VM, Azure Cosmos DB, and Azure Data Lake. All of these integration capabilities can be used to get a near-real-time copy of all or selected data from the CDS 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 Dynamics 365 Overview*, you learned about the various Dynamics 365 applications in regards 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 CDS 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 Dynamics 365 Finance and Supply Chain Management applications. Further details about this integration 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 might be necessary, 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*, CDS 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 CDS API to develop a batch outbound integration to retrieve larger datasets of CDS entity data and use it to integrate with other IT solutions.
- Use the *Web API* or *SOAP*-based CDS API to implement any kind of outbound or inbound integration solution, by calling any of the CRUD methods available in the CDS API.

With this, we have covered the typical backed integration patterns and solution approaches. Now, we can go ahead with the remaining integration scenarios, including Power Virtual Agent, AI Builder, and Power BI.

Presenting 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 in any case 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, which can be achieved in a low-code/no-code way. Both products cannot work standalone, so they must be integrated with other cloud services to benefit from those advanced capabilities.

Power Virtual Agent, as a chatbot solution, needs to be integrated into a publicly facing product to be available to customers. The product supports standard integration into the following 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 App portals:** The chatbot can be integrated using the same code snippet into an IFrame that's created in the Power Apps portal using the Power Apps portals Studio.
- **Teams:** The chatbot can be integrated into the Teams environment using a simple customization of the Teams environment with the Microsoft Teams App Studio.
- **Facebook:** The chatbot can be integrated into the Facebook Messenger on an organizational 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 Agent 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 most 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 *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 dashboard can be included in a model-driven app by a user as a personal dashboard. This integration can be performed directly by the model-driven app end user.
- **Canvas apps:** Power BI tiles can be added to a canvas app and vice versa, where canvas apps can be integrated into Power BI reports. This integration can be performed directly in the Power Apps Studio, or in the Power BI Desktop.

- **Power Apps portals/public website:** Power BI reports and dashboard can be added to Power Apps portal web pages once this capability is enabled in the portal settings. This integration can be performed directly in the Power Apps portals 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 entity 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 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 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 CDS entity 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 CDS API. Use the *client-side Web API* to make calls to CDS, 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. Frontend integration while implementing communication with third-party IT systems and solutions 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 error 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 CDS API require the use of the *primary key (GUID)* of the entity record to identify the record for update or deletion. However, in integration scenarios, the primary keys of the CDS 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 fields in the CDS entities (for example, customer number, product number, and case number) and configure those fields 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, which 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 CDS yet, but some are already stored in CDS 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 CDS API. An UPSERT request can automatically detect whether a record already exists in CDS. 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 CDS API. A batch request can contain up to 1,000 individual requests.
- For pure outbound integration scenarios, consider using the new CDS API endpoint known as **Tabular Data Stream (TDS)**. At the time of writing, this endpoint can only be used for reading data from the CDS database, but skilled SQL developers will quickly find it very useful for designing complex queries against CDS using the SQL language. The TDS endpoint is also very useful for checking CDS data using *SQL Server Management Studio*, or to use it for building Power BI reports and dashboards.

In this chapter, you learned about two similar products for frontend integration – *USD* and *UI flows* – and it is important to understand their differences. USD is an older product since it's been part of Dynamics 365 for years, while UI flows is rather new to Power Platform. But this is certainly not the only – and not the most important – difference. The main differentiator is the usage scenario. While USD is a tool for building agent desktop solutions for contact center and call center agents, the solution is completely used by *human users*. On the other hand, UI flows is used for autonomous integrations, *without* any human interaction.

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.
- They will consider an integration with Skype for Business, at least until a telephony integration with Teams is made available.
- They will further analyze the possible benefits of integrating with Microsoft 365 Groups, though an integration has not been planned yet.

Contoso Inc. have understood the various possibilities for integrating their Dynamics 365 solution with various cloud database services. They will further investigate which of the capabilities will best suit their needs. This will help them build a cloud data warehouse for collecting and consolidating data from all the relevant IT systems in the organization.

Frontend integration

Contoso Inc. have analyzed the frontend integration capabilities of CDS 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 criteria for using these capabilities is the best possible end user experience, so the solutions 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 UI flows' 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, and for which read-only access is sufficient, will be integrated using *virtual entities*.
- The preferred way for integrating data, which 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 has considered this as the most scalable and secure solution for Contoso Inc. They have evaluated their existing IT systems having APIs, and it seems that all of them can be integrated by using some of the existing public data connectors. For integrating on-premise IT systems, an On-Premise Data Gateway will be implemented in the most secure way.
- In case 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 entities 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 very specific cloud services in other Microsoft clouds. 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 integrations 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 deliver an empty solution. Rather, there are various existing systems, and data from these systems must be migrated to the solution. So, it is an inevitable part of every large enterprise implementation to identify all 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.

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 their 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 job. It is really important to understand the importance of data migration, the possible solution approaches, and the tools and techniques that can be used to implement it. In the following sections, you will learn about the different high-level approaches for importing data into a CDS-based application.

Learning about 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 Power Platform. 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 in such a way that it will be possible to trigger the initial data load process into CDS, as illustrated in the following diagram:

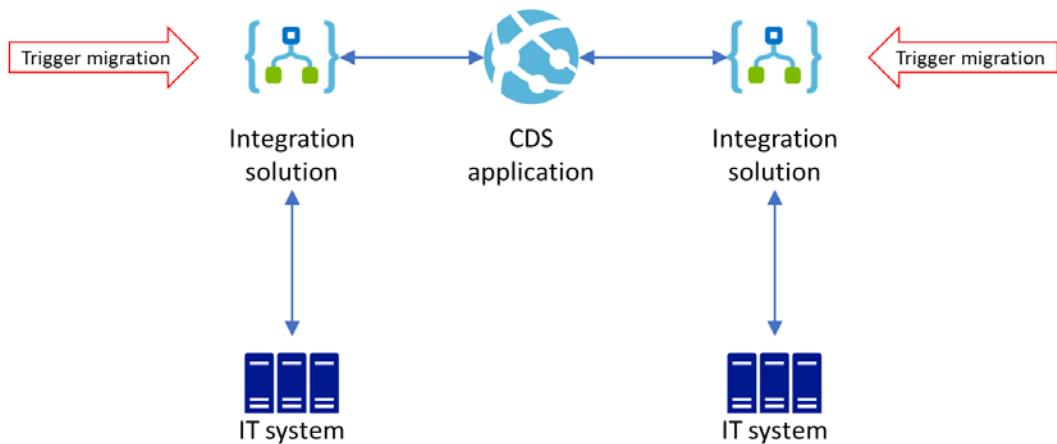


Figure 10.1 - Migration as part of integration

As we can see, there are two integrated IT systems containing data that should be imported into the **CDS** solution at the beginning of the live operation. 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. There can be a situation, where the overall data from several IT systems needs to be mutually consolidated first, in which case this approach would not work.

In the next section, we will describe another way to migrate data into CDS 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 this data directly. A *direct* data migration, using some of the available technologies, can be done like so:

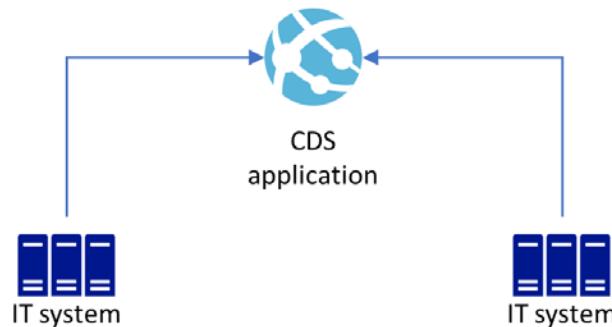


Figure 10.2 - Migrating consolidated data

As we can see, a direct migration from two 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 catalogue).
- 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, XML, CSV, and other formats, depending on the data import technology used.

However, for 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. For this scenario, 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 an **advanced migration** needs to be used, as illustrated in the following diagram:

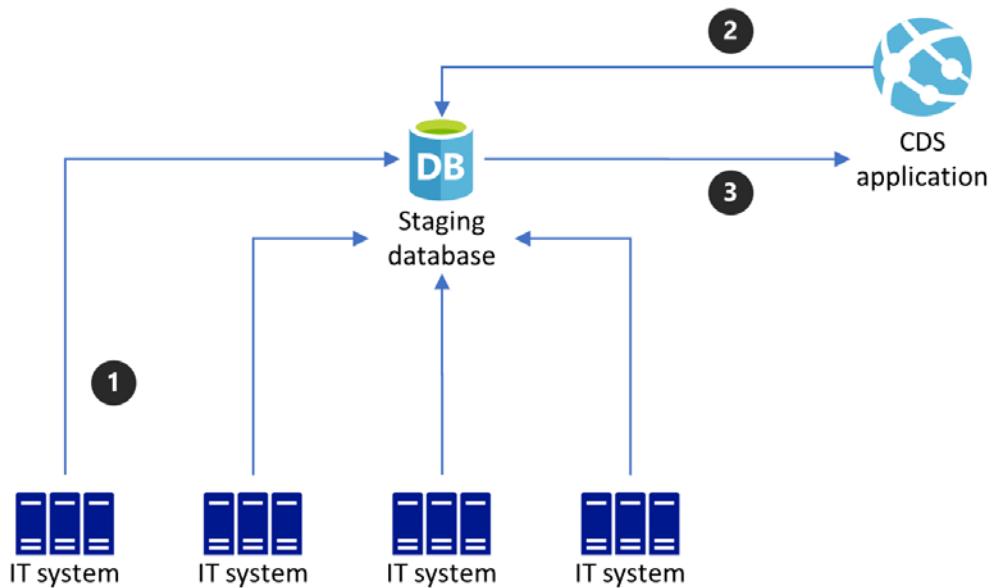


Figure 10.3 - Advanced migration

As we can see, advanced data migration consists of three major steps:

1. Load all the required data from all data sources in all possible technological formats into a **Staging database**. The **Staging database** is the consolidation platform for consolidating the data, restoring all required relationships, cleansing the data from possible duplicates, and so on.
2. Enhance the data in the **Staging Database** with CDS data to restore all required references and relationships. This is especially important when part of the solution data is created directly in the CDS database and a mapping needs to be restored before the final migration. Another main reason can be to restore the ownership of records with real existing CDS users. In this step, the data structure in the **Staging Database** must be prepared in a suitable form for direct import into CDS.
3. Direct data load into the CDS 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 CDS solution.

Understanding data migration tools and techniques

There are several ways to accomplish data migration to CDS. The CDS platform itself offers some out-of-the-box possibilities; however, these only cover the less complex migration scenarios. For the most complex scenarios, an external solution needs to be considered, as you'll learn in the following sections.

Doing manual entry

For the smallest data volumes where there's 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 CDS entity is to use the *Excel's export feature*. CDS applications provide a standard feature to export data from any entity 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 CDS. Regardless of whether there is already some data in the entity, this option can be used for *modification* of existing data or *import* of new record.

The procedure to follow to import data using Excel files consists of the following steps:

1. Prepare a system or personal view of the entity containing all fields required for import.
2. Select the prepared system or personal view in CDS and export data to Excel as a *Static Worksheet*.
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 CDS application.

The first three columns of the exported Excel file are hidden since they are used internally to identify existing records in CDS. An example of an Excel file is shown in the following screenshot:

A	B	C	D
1 (Do Not Modify) Account	(Do Not Modify) Row Checksum	(Do Not Modify)	Account Name
2 7bf0c988-20bd-e911-a82e-000d3ab0842d	ae4CChfHhwJl/feuhcqLvg7MI7NXp2T	24-10-19 20:34	A. Datum Corporation (sample)
3 6ff0c988-20bd-e911-a82e-000d3ab0842d	IG9c5d9iE9krNQTOWS1ij8LrChrtJ8vfH	24-10-19 20:34	Adventure Works (sample)
4 79f0c988-20bd-e911-a82e-000d3ab0842d	KTvh77HWMS8vdueLdbk155op3dDjG	24-10-19 20:34	Alpine Ski House (sample)
5 73f0c988-20bd-e911-a82e-000d3ab0842d	5tL8mRMVxrhyN8nvFdtTH20zxKQkt	24-10-19 20:34	Blue Yonder Airlines (sample)
6 75f0c988-20bd-e911-a82e-000d3ab0842d	6H0jQWmw46QwhvWHzk8omzDr	14-07-20 10:04	City Power & Light (sample)
7 7df0c988-20bd-e911-a82e-000d3ab0842d	WyZAnpcpVTI7AK3S1n5jaCfto9pYXP	24-10-19 20:34	Coho Winery (sample)
8 77f0c988-20bd-e911-a82e-000d3ab0842d	mL6nLbOKK6LZXUu92hvq+PckB/VcgB	24-10-19 20:34	Contoso Pharmaceuticals (sample)
9 71f0c988-20bd-e911-a82e-000d3ab0842d	I1kJMct2HGGBu4LhywWBEnPVJiRwB	24-10-19 20:34	Fabrikam, Inc. (sample)
10 6bf0c988-20bd-e911-a82e-000d3ab0842d	rQpOMrTEVmZLGeCPmYc9GiPwGUpj	19-07-20 06:38	Fourth Coffee (sample)
11 6df0c988-20bd-e911-a82e-000d3ab0842d	gvSGB0B3TDCEscAVury3I4WVHYmv5	02-07-20 11:35	Litware, Inc

Figure 10.4 - Exported Excel file from CDS

As we can see, when opening the first three *hidden* columns, we can see the **ID of the records**, a **Checksum**, and a **timestamp**.

Important note

It is very important not to modify the content of the first three columns in the Excel file in order to make a smooth reimport work properly.

Another way to use the Excel-based import feature is to download a *template* for data import from the **Data Management** area in the CDS settings. This template has the same structure as the one described in the preceding text but contains *ALL* the fields of the selected entity.

Importing data into CDS 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 enter the data in an appropriate structure and with good quality.
- It is possible to fill in the foreign key fields (lookup fields) with the exact values of existing records to ensure relations will be recreated.

Importing data using this approach can only be used to import into *one entity*; importing into several entities within a single import step is not possible. The next out-of-the-box data import option is to use the *data import wizard*, as described in the following section.

Using the data import wizard

If you require some more capabilities, such as importing data into several entities at once, you can use the **data import wizard**. This standard feature offers some more options for importing data, such as the following:

- The following data formats are supported: XML Spreadsheet 2003 (.XML), CSV files, TXT files, Excel files, 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 CDS for later import steps.
- The wizard allows you to import data into **several related entities** at once using the ZIP file option. In this case, import data for multiple CDS entities can be packed into a ZIP file and imported at once. During the import process, the relationship-relevant fields can be specified. This makes it possible to recreate the relationships during the import process.

The following image shows an example of the data mapping capability of the import wizard:

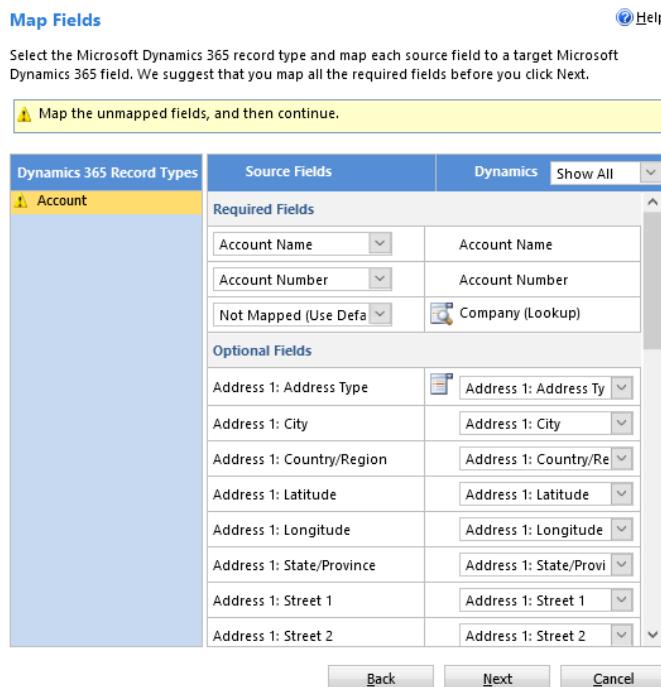


Figure 10.5 - Field mapping in the import wizard

As we can see, at the beginning of the mapping area, there are the **Required** entity fields, which must be mapped to valid data from the import file. After that the mapping of all other fields continues. In the last step of the import wizard, you 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 field mapping. There are the following file size limitations:

- Any file type containing data for a single entity 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 entities, the maximum file size is 32 MB.

The data import wizard's capability can also be used programmatically by using a collection of *CDS 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 CDS users

Important note

For more details about the programmatic use of the data import wizard, please refer to the product documentation: <https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/import-data>.

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 to another.

Introducing the Configuration Migration Tool

You were briefly introduced to the **Configuration Migration Tool** in *Chapter 4, Tool and Techniques*. In this section, we'll explain the capabilities of this tool in more detail with the help of the following diagram:

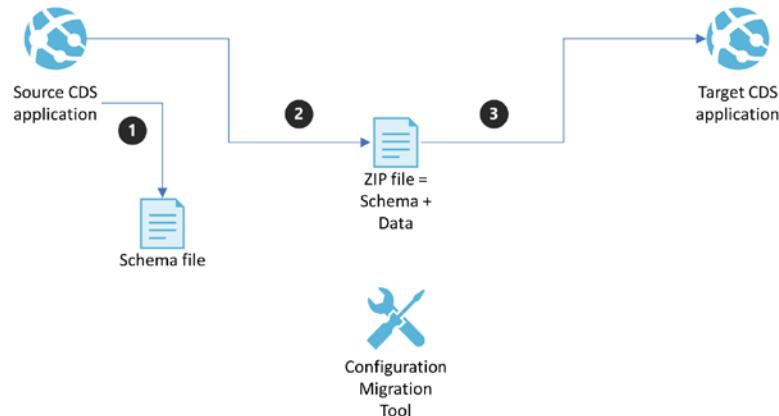


Figure 10.6 - Configuration Migration Tool capabilities

As we can see, the tool works in three steps:

1. In the first step, we select the **Source CDS** environment and create the migration **Schema** by selecting all the entities and fields 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 CDS 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 entity by entity using the *Excel file* approach.

The tool has the following capabilities:

- Filter records for migration.
- Preserve relationships between data in migrated entities.
- Validate the created schema for consistency.

- Move the datetime 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 CDS 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, Tools and Techniques*.

For more advanced data migration scenarios, we can consider using the out-of-the-box capability known as *Dataflows*, as explained in the following section.

Dataflows with Power Query

Dataflows is an advanced data migration tool that's directly available in the Power Platform *Maker Portal*. Dataflows can import data from a large number of different data sources, as illustrated in the following screenshot:

The screenshot shows a grid of icons representing various data sources. The columns are labeled: All categories, File, Database, Power Platform, Azure, Online services, and Other. A search bar is at the top right. The grid contains 24 items, each with an icon and a label: Excel File, Folder File, JSON File, PDF File; Parquet File, SharePoint folder File, Text/CSV File, XML File; Access Database, Amazon Redshift Database, Google BigQuery Database, IBM Db2 database Database; Impala Database, MySQL database Database, Oracle database Database, PostgreSQL database Database; Power Platform dataflows Power Platform, SAP BW Application Server Database, SAP BW Message Server Database, SAP HANA database Database; SQL Server database Database, Snowflake Database, Teradata database Database, Azure Blobs Azure; Azure Data Explorer (Kusto) Azure, Azure Data Lake Storage Gen2 Azure, Azure HDInsight Spark Azure, Azure SQL database Azure; Azure Synapse Analytics (SQ...) Azure, Azure Tables Azure, Salesforce objects Online services, Salesforce reports Online services; SharePoint Online list Online services, OData Other, Odbc Other, SharePoint list Other; Spark Other, Web API Other, Web page Other, Blank table Other.

Figure 10.7 - Dataflows data import sources

As we can see, many of the widely used databases 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 data to be imported for multiple entities.
- The tool offers advanced mapping and data transformation capabilities provided by the *Power Query* data transformation language.
- While importing data into CDS, the import can be performed for existing entities, as well as for entities 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 refreshes the CDS data on a regular basis.

Dataflows can be used as the preferred tool for every situation where there is data available in consolidated structures, ready to be imported into CDS 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-premise data source, the *On-Premise Data Gateway* product must be installed.

Dataflows can be also used to migrate data between two CDS environments. For this scenario, the dataflows are created in the target environment, and the **OData** data source which will be configured with the URL of the source CDS 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 fulfil very complex migration requirements, there is a need to reach out to an external toolset, as explained in the following section.

Using 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 CDS data migrations is **Microsoft SQL Server Integration Services (SSIS)**.

Important note

It is possible to use an on-premise Microsoft SQL Server instance, or an instance deployed in the cloud, within an **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 CDS, to achieve the smallest possible latency time when communicating between SSIS and the CDS API.

To make the life of the data migration developer easier, it is recommended to use a proper connector to CDS. Since Microsoft does not provide a dedicated native CDS 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 CDS, 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 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 *SQL Server Integration Services*. 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 CDS database. This helps us perform proper mapping between the data source and CDS. Using the new *TDS* endpoint makes it easier for the developer to work with the CDS database. It is possible to open both the staging database and the CDS database in the *Microsoft SQL Server Management Studio* in parallel, in order to work with the data, as illustrated in the following screenshot:

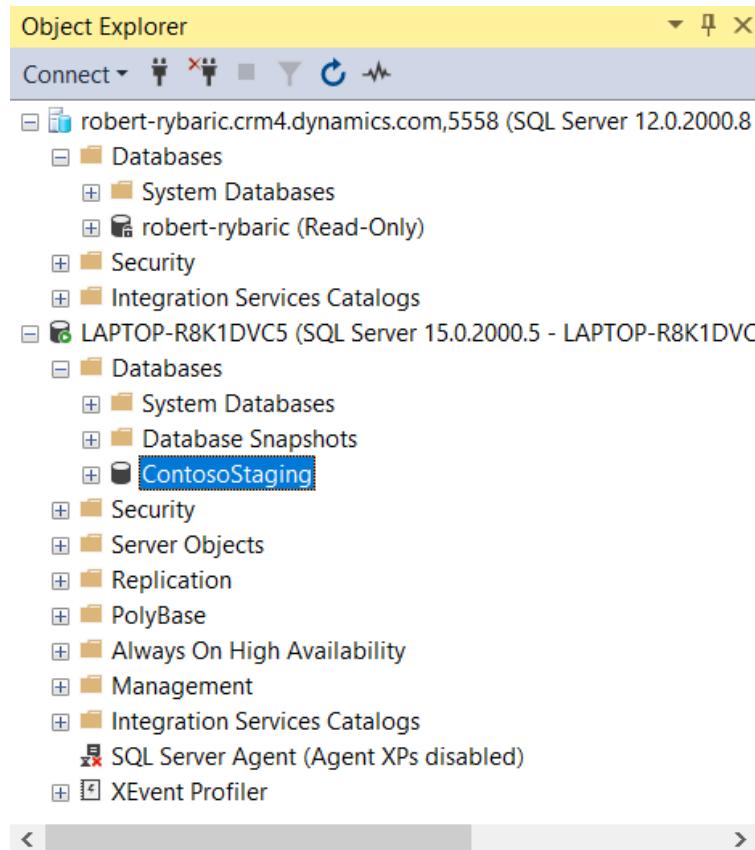


Figure 10.8 - CDS database opened along with local databases

As we can see , the CDS 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 CDS entity to be included in the data migration.

For actually importing CDS data into the staging database, the *KingswaySoft* adapter can be used within a SSIS task. For this, you need to configure the connection to CDS and to the staging database and then create a data flow with the appropriate mapping, as illustrated in the following screenshots:

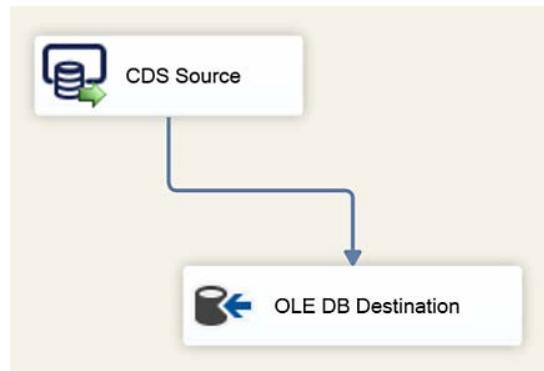


Figure 10.9 - SSIS data flow

As we can see, the solution consists of a 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:

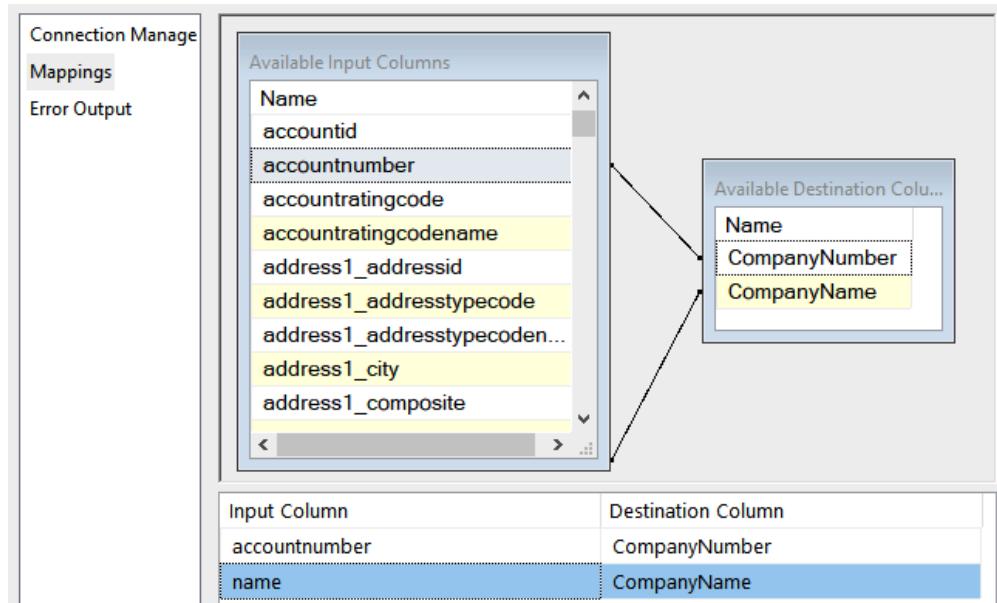


Figure 10.10 - SSIS data mapping

As we can see, the tool makes it possible to perform *data mapping* between the CDS 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 CDS 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 CDS 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 is of a different quality, into a set of data sources so that it can be imported into CDS. There is no single best way to do this; expertise in both the source data structures as well as in 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 CDS. For this, we can use the *KingswaySoft* adapter in the role of data flow destination, configured in a similar way to how the source role was. 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:

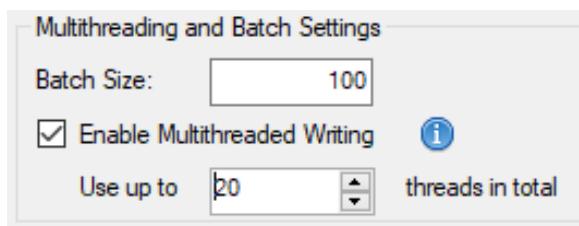


Figure 10.11 - Multi-threading and batch settings

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 CDS API, can be configured. It is important to choose appropriate values to prevent API limits 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.

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 each large enterprise 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 implementation project. This helps ensure the migration solution is ready at going-live time.

Scoping the migration

In a large Power Platform implementation, at the beginning, it is often required by the customer 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. **Scoping a migration** is therefore 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 must not be physically imported. It is sufficient to visualize them using solution approaches such as *virtual entities* or embedded *HTML web resources*. The best practice to decide the scope for the data migration is to classify the data entities 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 entity can probably be excluded from the physical migration.

Another part of scoping is to decide whether all the records from a selected data entity need to be migrated or that only newer records, such as those created in the last year or in the last 3 years, need to be migrated. Older records can be considered as 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 CDS. In industries such as retail, travel, banking, and others, every customer can generate from hundreds to thousands of individual retail transaction records. While the data itself is very interesting for a CDS solution, there is no reason to import that data physically into CDS due to the huge data volume. A much better solution would be to either visualize the data using one of the approaches mentioned in this chapter, 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 CDS solution. As you learned in *Chapter 3, Understanding Microsoft Power Platform Architecture*, the CDS comes with a certain default storage capacity that depends on the number of purchased Power Apps or Dynamics 365 product licenses. It is important to perform a high-level storage requirement analysis. This helps us assess whether the available storage space will be sufficient to accommodate all the data that's planned to be imported. At this point, it can be also decided 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 us assess whether all the data that's been planned to be migrated to CDS 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 migrated, or not allowed to be indirectly included in a relationship management solution such as CDS.

Understanding access issues

Data migration often needs various data sources from legacy solutions running in their 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 them 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 for 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 as possible of the technical documentation of the systems. 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 to not correspond to 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 any way to do that; or after the migration process in the CDS solution itself. However, experience has showed 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 CDS is the *ownership* of records. Records in every entity with the *user/team* ownership type must have an *owner*. The owner of the record is one of the most important data elements for the whole CDS 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 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, when possible, to 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 are not planned to be users of the CDS application. For this specific use case, the 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 identity, under which the migration solution will work. This identity will need to be provisioned in the CDS solution and equipped with all the necessary access rights. This is necessary to be able to write data to all planned CDS data entities and to perform all other required API calls. This identity will be the owner of every data record, for which no specific owner is explicitly defined.

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 CDS data structure. The mapping for simple data types such as **string**, **number**, **date**, and **time** is usually simple, but mapping for CDS-specific data types such as **option set**, **image**, **customer**, and **file** can be more challenging.

Data mapping generally consists of the following steps:

- Map the data on the **entity** level (decide which source data entity maps to which CDS entity)
- Map on the **fields** level (decide which fields from the source entity map to which fields in CDS)
- Map on the **field values** (this is specifically important for fields where the possible values are limited, such as option sets, lookup fields, and others)

One of the typical issues is to map the most often used data type of an option set correctly. As we know, the option set consists of an internal numerical value and a text representation of that value. For correct mapping, the option set values must be imported from CDS to be available for mapping.

Another important consideration is to understand that *datetime* fields are physically stored in the CDS database in **UTC** format. For performance optimization reasons, it is recommended to set the time zone of the user or service identity performing the migration to **UTC** as well. This will avoid time zone conversions during data load. It is also very important to understand the datetime formatting in the source IT systems, which must not necessarily be in the expected time zone. Here, a conversion would need to be performed.

Understanding record relationship issues

The CDS database requires that we create a consistent data model that includes all the relationships between entities. A relationship behavior between entities cannot be implemented without creating a physical relationship. However, data in source IT systems is not always stored in database systems and respects the same level of consistency between related data entities. It is common, that physical relationships are not created, and then the data tends to miss the required full relational consistency. Simply put, parts of the records in expectedly related entities do not have the connection between them established. This issue is extremely hard to resolve and in most cases, these records can just be excluded from the migration process.

Understanding business process flows issues

A very specific situation can occur, when an assignment of certain **business process flows (BPF)** to imported records is required. This can happen for records in the typical main business entities, such as the opportunity for sales management or case for customer support management. The real challenge is not assigning the BPF itself, but assigning the required stage. When creating a record in CDS via the CDS API, it is possible to assign a BPF, but only in the first BPF stage. Moving to the required subsequent stages can be done only stage-wise. For example, should an opportunity record be assigned to the fourth stage of a certain BPF, upon creation, the BPF will always be in the first stage. The move to the fourth stage would need to be performed as a BPF stage-change request, which would need to be triggered three times.

Understanding record status issues

Every CDS record has *status* and *status reason* fields by default. The status field 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 in some of the inactive statuses and then make changes to the records and/or create records in the underlying entities. As an example, we could consider importing old opportunities that are mostly closed together with opportunity products. The correct sequence of import steps would need to be as follows:

1. Import an opportunity record as open.
2. Make all required updates on 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 entities pretty complex. This is because it consists of many individual steps in order to respect all the specifics of the CDS business entities.

Setting certain system fields

Every business record in CDS has a whole bunch of system datetime fields indicating the creation date of the record, as well as the date of last record modification. It is important to analyze the default behavior (fields will be set to the current date of data migration) and decide whether this is a 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 field in every CDS entity that can be used instead of the standard **created on** field. The name of this field is `overriddencreatedon`. Using this field in the migration mapping process will set the **created on** field 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 frequent requirements for data migrations is to migrate not just relational data, but also documents of any type – related or even unrelated to relational data records. Importing documents can be done either by importing directly into the CDS 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 CDS 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 CDS entities. In such a complex case, the order of the import steps is extremely important. 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, containing also a lot of custom entities. 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 using record creation steps only. 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. Update step to set 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 *SQL Server Integration Services 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, CDS 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 SQL Azure database is available. Saying that, it is very important to consider the overall data migration duration when planning the final migration before going live. The CDS endpoints, Web API, and SOAP provide certain transaction performance that is also limited by API limits, as described in the next section. To achieve the best possible migration performance, all those 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, a full test execution is necessary.

A specific topic to consider when planning a data migration and preparing the design is to decide which automations can be, or must be, disabled during the import's execution. A complex CDS solution can have a lot of various automations implemented (workflows, PlugIns, auditing, duplicate detection, Power Automate flows, and others). Some of those automations must be switched off due to business requirements since they should only be executed in normal operation. Some of the automations 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 Microsoft Power Platform Architecture*, the CDS API has some protective limits regarding how many requests a single process can trigger against the API within a defined time period. This is specifically unfortunate for large data migration projects with an expected high number of records to be created.

Important note

For further details about handling the API limits within a data migration solution, please refer to the product documentation: <https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/api-limits>.

One possible option to overcome this limitation is to request to lift these limitations 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 long, taking the usual speed of the CDS 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, 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 steps and each step 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 step.

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 the 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 is very diverse and complex, and that the data migration process will be not simple. The following are their key findings:

- Within the whole organization, there is 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 option 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 *CDS server-side synchronization* capability to bring the contacts into CDS. They will train the end users to synchronize their most important and high-quality contacts into CDS using the *Dynamics 365 App for Outlook*.
- The product catalogue is part of 10 various global or local IT systems. There will be a need to consolidate this data into the CDS product catalogue.
- Contoso Inc. and their regional subsidiaries are using more than 10 sales management solutions to track leads and opportunities. This data will need to be consolidated and migrated into CDS. 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 the active and closed past service cases, along with some information of type knowledge articles. This data must also be consolidated and imported into the CDS solution. Since Contoso Inc. is eagerly waiting for the chance to analyze the efficiency of the overall customer support organization, they 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 CDS 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. is heavily using documents in the context of their 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 into CDS in an integrated SharePoint repository.

Besides the most important data entities that were identified during the analysis, they documented a large number of various additional and configuration data, as well as other, smaller pieces of 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, based on a later decision, imported into CDS.

The analysis has clearly indicated that the data migration process will be complex and needs to use proper tooling. That's why they decided to use the recommended approach of *SQL Server Integration Services* 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:

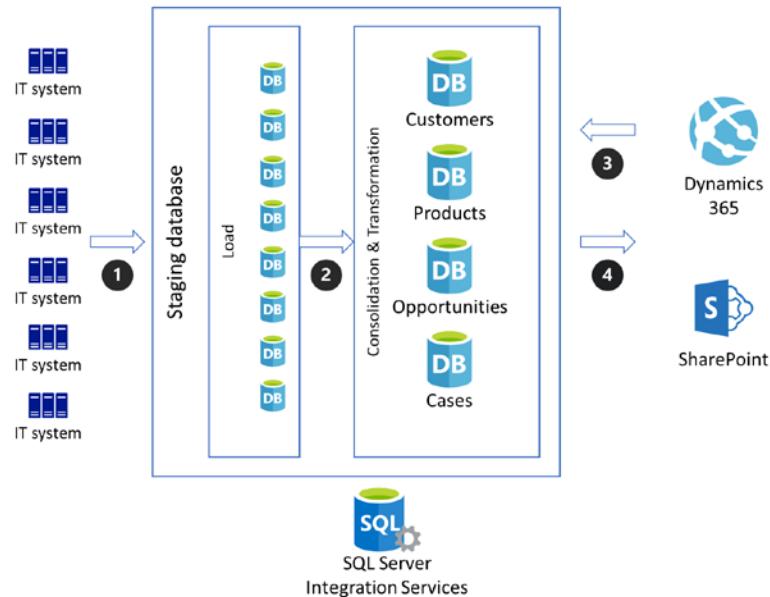


Figure 10.12 - Contoso Inc. data migration setup

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 import 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 they have the most important data entities 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 required data 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 CDS foreign keys in the target importable database tables. For every such requirement, the content of a CDS entity 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 CDS, 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 the single-step and the 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 for all included IT systems 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 project can be easy and need 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. Due to this, you learned about the different tools you can use for migrating data, as well as their usage scenarios.

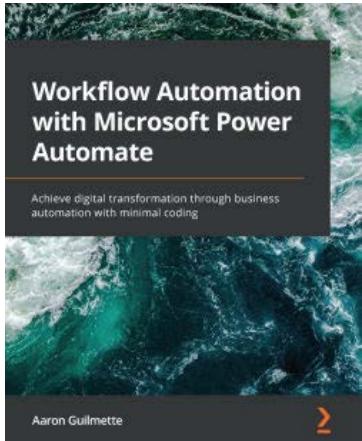
In the last part of this chapter, you came across a lot of typical challenges and ways to mitigate them in order to be successful and deliver a smooth data migration project.

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!

Other Books You May Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:

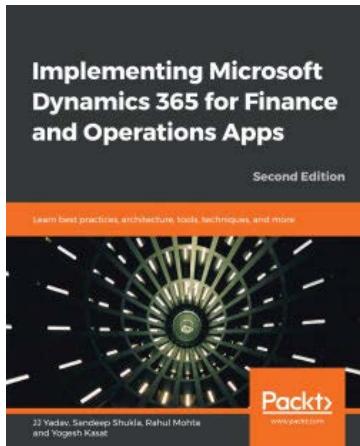


Workflow Automation with Microsoft Power Automate

Aaron Guilmette

ISBN: 978-1-83921-379-3

- Get to grips with the building blocks of Power Automate, its services, and core capabilities
- Explore connectors in Power Automate to automate email workflows
- Discover how to create a flow for copying files between two cloud services
- Understand the business process, connectors, and actions for creating approval flows
- Use flows to save responses submitted to a database through Microsoft Forms
- Find out how to integrate Power Automate with Microsoft Teams



Implementing Microsoft Dynamics 365 for Finance and Operations Apps – Second Edition

Jila Jeet Yadav, Rahul Mohta, Sandeep Shukla, and Yogesh Kasat

ISBN: 978-1-78995-084-7

- Understand the architecture of Dynamics 365 for Finance and Operations Apps
- Implement Dynamics with confidence to manage finances in your business
- Get up to speed with different methodologies and support cycles of the Microsoft Dynamics architecture
- Explore best practices to analyze the requirements of your business
- Understand the technique of data migration from legacy systems
- Leverage the capabilities of Power BI to make informed business decisions
- Manage all your upgrades through One Version service updates

Leave a review - let other readers know what you think

Please share your thoughts on this book with others by leaving a review on the site that you bought it from. If you purchased the book from Amazon, please leave us an honest review on this book's Amazon page. This is vital so that other potential readers can see and use your unbiased opinion to make purchasing decisions, we can understand what our customers think about our products, and our authors can see your feedback on the title that they have worked with Packt to create. It will only take a few minutes of your time, but is valuable to other potential customers, our authors, and Packt. Thank you!

Index

A

AAD Sync 281
administration centers
 about 81
 Microsoft 365 Admin Center 82
 Power Apps analytics 84
 Power Automate analytics, areas 84
 Power Platform Admin Center 82, 83
adoption and change management (ACM)
 about 213
 responsibilities 213
agent desktop 332
agile model
 about 189
 features 189
AI Builder
 about 129, 383
 characteristics 29
 integration, ways 384
Ajax 366
API administration
 about 89
 Microsoft 365 administration 89, 90
 Power Apps administration 90
 Power BI administration 91, 92
 reference link 89

application lifecycle management (ALM)
 about 139, 145, 146
 best practices 171
 component libraries 174
 components, sharing 174
 decisions 178
 environment complexity 146, 147
 for Power BI 168
 for Power Platform-based solution 148
 for solution components 171
 implementing, with Contoso Inc. 146
 multiple solutions, using 172, 173
 Power BI best practices 176
 Power Platform solution
 complexity 147, 148
 segmentation, using 174, 175
 solution best practices 171
 solution publishers best practices 175
 solution structuring 172
 source control, using 175
application lifecycle management (ALM), solution best practices
 general practices 171
 unmanaged solution, versus
 managed solution 172
application life cycle management tools

- presenting 139
- architectural best practices
 - administration and monitoring 110
 - environment regions 109, 110
 - environment setup 101, 102
 - environments, testing/developing 98
 - presenting 97
 - single tenants or multiple tenants 97
 - unsupported integration topology 99
- architecture board 212
- auditing technologies
 - CDS auditing 96
 - Office 365 Activity Logging 95
- authentication
 - about 234, 237
 - Azure Active Directory guest users 253
 - of external users 253, 254, 255
 - service authentication, for
 - internal users 247
 - versus authorization 234
- authentication features, for internal users
 - about 244
 - ADFS claim rules 245
 - conditional access 244
 - cross-tenant inbound
 - restriction 246, 247
 - cross-tenant outbound restriction 247
 - multi-factor authentication (MFA) 245
 - seamless sign-on options 246
- authentication governance,
 - for internal users
 - about 251
 - CDS session governance 252, 253
 - CDS user accounts provisioning
 - governance 251, 252
- authorization
 - about 235, 256
 - group teams 261
- in CDS 257
- in model-driven apps 257, 262-264
- in Power Apps portals 271, 272
- in Power BI 270
- in Power Platform 256, 257
- authorization, in Canvas Apps
 - about 268
 - app authorization 268
 - connectors authorization 269
 - possibilities, for sharing 268
 - sharing types 268
- authorization, in Power Automate
 - about 269
 - background flows authorization 269
 - interactive flows authorization 270
- authorization, in Power BI
 - DirectQuery dataset 270
 - import dataset 270
- authorization options, CDS-based solution
 - access teams 266
 - field-level security 267
 - hierarchy security 265
 - hierarchy security, implementing 265
 - record sharing 266
 - user interface security 267
- Azure Active Directory (ADD) 51, 234
- Azure Active Directory B2C
 - (Business to Consumer) 51
- Azure Active Directory integration 357
- Azure API Management
 - about 52
 - integrating with 362
- Azure Application Insights 96
- Azure Blob Storage
 - about 54
 - integrating with 357, 358

Azure Cosmos DB
about 53
integrating with 360, 361

Azure Data Lake
integrating with 359, 360

Azure Data Lake Storage Gen2 54

Azure DevOps
about 55, 140, 196
build tools, for administration
capabilities 94
capabilities 141
establishing 176
for Power Platform 161
used, for administration 94

Azure DevOps, using with Power Platform build tools
about 163
pipeline, versus release 167
solution, committing to source control 163, 164
solution, distributing between development environments 164-166
solution, distributing out of development 166, 167

Azure Event Hub 52

Azure Event Hub integration 357

Azure Functions 52

Azure IoT Central 54, 348

Azure IoT Hub 54, 348

Azure Key Vault 54

Azure Log Analytics 96

Azure Log Analytics Data Collector 96

Azure Logic Apps
about 52
integrating with 361

Azure Monitor 55

Azure Service Bus 51

Azure Service Bus integration 357

Azure SQL 53

Azure Virtual Machine 403

B

backend components
testing 138

backend integration patterns,
and solution approaches
about 368

custom backend integrations 382

data integration pattern 379

publish-subscribe pattern 374-377

relay pattern 371-373

remote procedure call (RPC)
pattern 368-370

request-callback pattern 377

best practices, for environment setup
production environments 106

Bring-Your-Own-Key (BYOK)
87, 185, 274

business consultant
responsibilities 207

business process flows (BPFs)
295, 307, 412

C

canvas apps
authorization 268
capabilities 25
characteristics 333
creating 26
customizing, solutions 334
embedding 318, 319

canvas apps and Power Automate
customization 333

- canvas apps and Power
 - Automate extensibility
 - custom connectors, building 334, 335
 - presenting 333
- Canvas Apps Designer Studio
 - about 125
 - capabilities 126
- capabilities, for Power Automate flows
 - actions 72
 - tables 72
 - triggers 72
- capacity restrictions, CDS
 - about 69
 - API limits 71
 - request limits and allocations 71
 - storage capacity limits 69, 70
- CDS
 - integrating, with Exchange 349
- CDS authorization
 - basics 257
 - business units 259
 - entity-based permissions,
 - permission levels 260
 - permission levels 260
 - record ownership 259
 - record ownership, types 259
 - security roles 259, 260
 - teams, types 258, 259
 - users and teams 258
 - users, types 258
- CDS automation 303. *See also* processes
- CDS business process flows (BPFs) 306
- CDS business rules 304
- CDS client-side extensibility
 - about 309
 - canvas apps, embedding 318, 319
- Power Apps Component
 - Framework (PCF) 311, 312
- standard custom controls 309, 311
- CDS client-side extensibility, entity-specific command bars
- entity form command bar 316
- entity sub-grid command bar 316
- main grid command bar 316
- CDS client-side extensibility, form scripting
- Field OnChange event 315
- Form OnLoad event 314
- Form OnSave event 315
- CDS client-side extensibility, web resources
 - about 313
 - command bar/ribbon extensibility 315-317
 - form scripting 314, 315
 - web components, embedding 314
- CDS client-side extensibility, web resources types
 - code web resources 313
 - data web resources 313
 - graphical web resources 313
 - user interface web resources 313
- CDS client-side interface
 - extensibility 336, 337
- CDS content-based security
 - about 276
 - entity form switching, using 277
 - server-side event handlers, using 277
- CDS custom actions 305
- CDS customization artifacts
 - Hierarchy Settings 302
 - Reports 302
- CDS data modeling 296-298
- CDS environment
 - inactivity timeout 253
 - session timeout 252

- CDS extensibility
 - presenting 292
- CDS security roles
 - about 274
 - layering 275, 276
 - layering, benefits 276
 - modifying 275
- CDS server-side extensibility
 - about 320, 338
 - Azure Event Hub integration 328
 - Azure Service Bus integration 325-328
 - CDS API selection 338
 - custom workflow actions 325
 - extensibility and automation
 - options 339, 340
 - external applications, building 329
 - performance impact 340, 341
 - plugin event handlers 322-325
 - Web Hook integration 328, 329
- CDS server-side extensibility,
 - API interface types
 - about 320-322
 - Deployment Web Service 320
 - Discovery Web Service 320
 - Online Management API 321
 - Organization Data Service 320
 - Organization Service 320
 - Tabular Data Stream 321
 - Web API 320
- CDS server-side extensibility,
 - pipeline stages
 - difference 324
 - PostOperation 324
 - PreOperation 324
 - PreValidation 324
- CDS session governance 252, 253
- CDS solution 393
- CDS standard customization
 - about 293-295
 - mobile apps, designing 303
 - model-driven applications,
 - designing 302, 303
- CDS user accounts provisioning
 - governance 251, 252
- CDS user interface design
 - about 298
 - entity charts 300
 - entity forms 298
 - entity-specific and entity-independent dashboards 301
 - entity views 299
- CDS user interface design,
 - entity forms types
 - Card Form 299
 - Main Form 298
 - Quick Create Form 299
 - Quick View Form 299
- CDS user interface design, entity-specific and entity-independent dashboards types
 - interactive experience dashboard 301
 - standard dashboard 301
- CDS user interface design,
 - entity views types
 - calendar views 300
 - editable views 300
- CDS workflows 305
- centralized customization 164
- Charts
 - managed properties 153
- citizen developer
 - about 118
 - versus IT pro developer 119
- Clone a patch 158
- Clone solution 158

- cloud identity approach
 - about 238, 243
 - capabilities 239
 - implementing 238
- Code generation tool 136
- combined model 191
- command bar 123
- Common Data Model (CDM) 23
- Common Data Service (CDS)
 - about 64, 147, 251
 - capacity restrictions 69
 - components 23
 - storage types 69
- Common Data Service (CDS)
 - authentication
 - about 247, 257
 - OAuth authentication 249, 250
 - Office 365 authentication 248
- Common Data Service (CDS),
 - storage types
 - database 69
 - file 69
 - log 69
- compliance 273, 408
- Compliance Manager
 - about 273
 - reference link 273
- concepts, for managing authorization
 - in cloud services
- CDS database applications 236
- Exchange Online 236
- Power Apps and Power Automate flows 236
- Power BI 236
- SharePoint Online 236
- conditional access feature 244, 245
- configuration and customization tools
 - about 119
- AI Builder 129
- Canvas Apps Designer Studio 125, 126
- dataflows designer 130
- Forms Pro Designer 133
- ISV Studio 134
- Microsoft AppSource 133
- PowerApps Portal Studio 124
- Power Automate designer 126, 128
- Power Virtual Agents designer 128
- configuration migration tool 140, 400
- configuration page 152
- Continuous Deployment (CD) 167
- Continuous Integration (CI) 167
- Contoso Inc.
 - about 20, 46, 118, 392
 - Authentication and single sign-on, using 55
- Azure DevOps, using 56
- Azure Monitor, using 56
- citizen developers, enabling 143
- core project team, enabling 142, 143
- IoT integration, using 56
- Microsoft 365, using 49
- Microsoft Azure, using 55
- Microsoft Cloud App Security, using 50
- Microsoft Excel, using 50
- Microsoft Exchange, using 49
- Microsoft Intune, using 50
- Microsoft OneDrive for Business, using 50
- Microsoft OneNote, using 50
- Microsoft Outlook, using 50
- Microsoft SharePoint, using 50
- Microsoft Teams, using 50
- Microsoft Word, using 50
- Power Platform integration,
 - designing 346
- Power Platform integration, using 55

- Power Platform solution, designing 290
preparing, implementation project 182
used, for implementing ALM 146
- Contoso Inc. ALM
strategy 176
- Contoso Inc. Power Platform architecture
about 111
clients 114
environments 113, 114
environments, purposes 114
tenant structure 112
user groups and licensing 115
- Contoso Inc. Power Platform
commitment 42, 44
- Contoso Inc. Power Platform
integration design
about 387
backend integration 389
frontend integration 388
integration, with Microsoft 365
and Microsoft Azure 388
- Contoso Inc. Power Platform
solution design 341
- Contoso Inc. Power Platform
solution design decisions
about 343
automations 342
client-side extensibility 342
model-driven apps 341
server-side extensibility and
integrations 343
- Contoso Inc. project team
workplace setup 142
- Contoso Inc. security architecture
about 285
Active Directory integration 285
Common Data Service 285, 286
Data Loss Prevention (DLP) policies 285
- security decisions 286
Create, Read, Update, and
Delete (CRUD) 320
- CRMRestBuilder
reference link 137
- cross-tenant restrictions 246, 247
- custom backend integrations 382
- custom connector
developing and configuring,
steps 334, 335
- custom development tools
CRMRestBuilder 137
- NuGet developer tools and
assemblies 136
- Postman 137
- Power Apps Command-Line
Interface (CLI) 135
- Power Platform extensions,
for Visual Studio 136
- presenting 134
- Visual Studio 134
- Visual Studio Code 135
- XrmToolBox 137
- Customer Cloud structure,
Power Platform
about 64, 65
app registrations 66
group management 66
license management 65
Office 365 Activity Logging 66
user management 65
- ## D
- dashboards
about 366
managed properties 153
- Dataflows

- about 401
- with Power Query 401, 402
- dataflows designer
 - capabilities 130
- data import wizard 398
- data integration pattern
 - about 379
 - integration, between Dynamics 365 CRM and ERP 382
 - standard Azure data integrations 381
 - virtual entities, using 380, 381
- Data Loss Prevention (DLP) policies
 - about 285
 - connectors, placing into
 - blocked group 73
 - connectors, placing into
 - business data group 73
 - connectors, placing into
 - non-business data group 73
 - scope levels 73
- data mapping
 - on entity level 411
 - on fields level 411
 - on fields values 411
- data migration
 - about 392
 - advanced technique 394, 395
 - Configuration Migration Tool 400, 401
 - consolidated data 394
 - Dataflows, with Power Query 401, 402
 - design 416-419
 - learning, as integration 392, 393
 - overview 392
 - planning 392
 - SQL Server Integration Services, using 403
 - data migration, challenges and best practices
- about 407
- access issues 408
- API limits, resolving 415
- business process flows (BPF) 412
- complex data migration
 - process 413, 414
- compliance 408
- data migration, automating 414
- data migration, executing 416
- data migration, performance 415
- data, verifying by customer 416
- documents, migrating 413
- encoding issues 410
- lack of documentation 409
- lack of knowledge 409
- mapping issues 411
- planning and effort estimation 407
- record ownership issues 410
- record relationship issues 411
- record status issues 412
- scoping 407, 408
- source data quality 409, 410
- storage impact 408
- system fields, setting 413
- data migration, tools and techniques
 - about 396
 - data import wizard, using 398, 399
 - Excel files, using 396, 397
 - manual data entry 396
- data protection 273
- Data Service and model-driven apps tools
 - about 119
 - Maker Portal 120, 121
 - Model-driven app designer 121, 122
 - Model-driven app Sitemap
 - designer 122, 123
 - XrmToolBox 123
- default solution 153

desktop clients
about 76
browser client 76
Dynamics 365 App for Outlook 77
Dynamics 365 App for Outlook,
capabilities 77
Dynamics 365 App for
Outlook, features 77
Unified Service Desk (USD) 78

development environments
about 102
complex testing 106
complex testing, purpose 106
compositions 102
development team, and multiple
workstream 105
development team, and single
workstream 104
simple environments 103
simple environments, uses 103

distributed customization 164

dual-write 382

Dynamics 365 applications
integrating, with Microsoft
OneNote 353
integrating, with Microsoft
Teams 351, 352
integrating, with Microsoft Yammer 355
integrating, with OneDrive
for Business 350, 351
integrating, with SharePoint 350
integrating, with Skype/Skype
for Business 354, 355

Dynamics 365 Connected
Field Service 348

Dynamics 365 Marketing 348

Dynamics 365 Sales 348

E

Email Server Profile 349
enterprise architecture (EA) 212
Entities
managed properties 154
entity forms 365
entity views 365
environment complexity 146, 147
environment tasks 162, 163
environment variables
about 150
data types 150
Exchange
CDS, integrating with 349
extensibility
overview 290
extensibility, levels
configuration 291
custom development 292
customization 292
standard capabilities 291
external applications, CDS server-
side extensibility
alternate clients and portals 330
batch jobs 329
mobile clients 330
extract, transform, load (ETL)
about 379
approach 394

F

FakeItEasy
URL 138

FakeXrmEasy
URL 138

federation approach
about 242, 244
considerations 243
implementing 242, 243
Fields
managed properties 153
Field Service Mobile app 80
final migration process 416
final testing
activities 224
first-party application
updates 160
Forms
managed properties 153
Forms Pro
capabilities 29
Forms Pro Designer
about 133
capabilities 133
frontend integration patterns,
and solution approaches
about 363
CDS content, embedding into third-
party containers 365, 366
event-driven integration 366, 367
on-demand frontend
integration 366, 367
third-party content, embedding
into CDS 363-365
UI Flows, using 367
Unified Service Desk, using 368

G

general availability (GA) 108

H

hierarchy security
approaches 265
Hosted Application Toolkit (HAT) 368
HTTP Data Collector API 97

I

identity and access management (IAM)
using 281, 282
identity and authentication
solution, for internal users
about 238
cloud identity approach 238, 239
federation approach 242, 243
pass-through authentication
approach 240, 241
password hash synchronization
approach 239, 240
IFrame 363
implicit Dynamics 365 integrations 347
independent software vendor (ISV) 134
Infrastructure as a Service (IaaS) 51
initial analysis
tasks 220
initial data load 392
integrated development
environment (IDE) 134
integration possibilities, for model-
driven apps integration with
Application Insights
client-side integration 96
server-side integration 96
integration possibilities, Power
Virtual Agent
reference link 384

ISV Studio
about 134
reference link 134
iterations overlap 190
iterative analysis
activities 221
iterative design 222, 223
iterative development
activities 223
iterative model
about 190, 191
features 190
iterative testing
activities 224
IT lead
responsibilities 211
IT pro developer
about 118
versus citizen developer 119
IT security
authentication, versus authorization 234
compliance 234
overview 234
privacy and data protection 234
security 234

J

jQuery 366

K

key users
responsibilities 210

L

lead architect
responsibilities 205
Life Cycle Services (LCS) 382
listener 325

M

Maker Portal 120, 121
managed properties
about 153
of Charts 153
of Dashboards 153
of Entities 154
of Fields 153
of Forms 153
of Views 153
managed solution 152
management connectors, Power Automate
Approvals connector 92
Azure Active Directory connector 92
Microsoft Forms connector 92
Office 365 Users connector 92
Power Apps for Admins connector 92
Power Apps for App Makers
connector 92
Power Automate Management
connector 93
Power BI connector 93
Power Platform for Admins
connector 92
used, for monitoring tasks 94
mass integration 379
mature security model 234
Microsoft 365
about 46
integrations 348

- overview 56
- Power Platform, integrating with 347
- Microsoft 365 Admin Center
 - about 82
 - activities 82
 - analytics areas 82
 - URL 82
- Microsoft 365 Business 57
- Microsoft 365 Education (A3 and A5) 57
- Microsoft 365 Enterprise (F3, E1, E3, E5) 56
- Microsoft 365 Groups
 - integrating with 356
- Microsoft AppSource
 - URL 133
- Microsoft Authentication Library (MSAL), for android
 - reference link 330
- Microsoft Azure
 - about 51
 - Azure Active Directory (AAD) 51
 - Azure API Management 52
 - Azure Blob Storage 54
 - Azure Cosmos DB 53
 - Azure Data Lake Storage Gen2 54
 - Azure DevOps 55
 - Azure Event Hub 52
 - Azure Functions 52
 - Azure IoT Central 54
 - Azure IoT Hub 54
 - Azure Key Vault 54
 - Azure Logic Apps 52
 - Azure Monitor 55
 - Azure Service Bus 51
 - Azure SQL 53
 - integrations 356
 - Power Platform, integrating with 347
- Microsoft Azure licensing
 - about 56, 57
 - overview 56
- Microsoft Cloud App Security 49
- Microsoft cloud authentication
 - and authorization
 - about 236
 - authorization, granting 235, 237
 - fundamentals 235
 - licenses, assigning 235, 237
 - provision user identity, provisioning 236
 - user identity, provisioning 235
- Microsoft Dynamics 365
 - Business Central app
 - capabilities 34
 - Microsoft Dynamics 365 Commerce app
 - capabilities 34
 - Microsoft Dynamics 365
 - Connected Store 37
 - Microsoft Dynamics 365 CRM
 - applications 30
 - Microsoft Dynamics 365
 - Customer Insights
 - capabilities 35
 - Microsoft Dynamics 365
 - Customer Service app
 - capabilities 31
 - Microsoft Dynamics 365 Customer Service Insights 36
 - Microsoft Dynamics 365 ERP
 - applications 32
 - Microsoft Dynamics 365 Field Service app
 - capabilities 31
 - Microsoft Dynamics 365 Finance app
 - capabilities 33
 - Microsoft Dynamics 365
 - Fraud Protection 37
 - Microsoft Dynamics 365 Guides 38

- Microsoft Dynamics 365 high-level architecture 20
- Microsoft Dynamics 365 high-level architecture, components about 22
- AI Builder 28, 29
 - canvas apps 25, 26
 - Common Data Model (CDM) 23
 - Common Data Service (CDS) 23
 - Forms Pro 29
 - model-driven apps 24
 - On-Premises Data Gateway 28
 - Power Apps portals 29
 - Power Automate 26
 - Power BI 28
 - Power Virtual Agents 27
- Microsoft Dynamics 365
- Human Resources app capabilities 34
 - Microsoft Dynamics 365 Import Tool 38
 - Microsoft Dynamics 365 Marketing app capabilities 30
 - Microsoft Dynamics 365 Product Insights 37
 - Microsoft Dynamics 365 Product Visualize 38
 - Microsoft Dynamics 365 Project Operations app capabilities 32
 - Microsoft Dynamics 365 Remote Assist 37
 - Microsoft Dynamics 365 Sales app capabilities 30
 - Microsoft Dynamics 365 Sales Insights 36
 - Microsoft Dynamics 365 Supply Chain Management app capabilities 33
- Microsoft Dynamics 365 Unified Service Desk 38
- Microsoft Enterprise Mobility + Security overviewing 48
- System Center Configuration Manager 49
- Microsoft Excel 48
- Microsoft Exchange 47
- Microsoft Graph API reference link 89
using 89
- Microsoft Intune 49, 114
- Microsoft Office 365 46
- Microsoft Office 365 Groups 48
- Microsoft OneDrive for Business 47
- Microsoft OneNote
- about 47
 - Dynamics 365 applications, integrating with 353
- Microsoft Outlook 48
- Microsoft Power Platform
- about 20
 - complex marketing management 42
 - first-party applications, updates 160
 - licensing overview 38, 40, 41
 - simple relationship management 41
 - updates 160
 - updating procedures 160
- Microsoft Privacy Statement
- about 273
 - reference link 273
- Microsoft Project desktop client 196
- Microsoft Project for the Web 196
- Microsoft Project Online 196
- Microsoft Service Trust Portal
- about 273
 - reference link 273
- Microsoft SharePoint 47

- Microsoft SQL Server Integration Services (SSIS) 403
- Microsoft SQL Server Management Studio 404
- Microsoft Teams
- about 47
 - Dynamics 365 applications, integrating with 351, 352
- Microsoft Trust Center
- about 273
 - reference link 273
- Microsoft Word 48
- Microsoft Yammer
- about 48
 - Dynamics 365 applications, integrating with 355
- mobile application management (MAM) 49
- mobile apps
- designing 303
- mobile clients
- about 80
 - Dynamics 365 for Phones and Dynamics 365 for Tablets 80
 - Power Apps mobile player 80
 - Power Automate mobile app 81
 - Power BI app 81
 - unified mobile client app 80
- mobile device management (MDM) 49
- model-driven app
- creating 25
- Model-driven app designer
- about 121
 - capabilities 122
- model-driven applications
- designing 302, 303
- model-driven applications, parts
- site map 294
- subset of CDS 294
- model-driven apps
- authorization 257
 - capabilities 24
 - components 24
- model-driven apps extensibility
- presenting 292
- Model-driven app Sitemap
- designer 122, 123
- Multi-factor authentication (MFA) 239
- ## N
- network traffic analyzers
- using 139
- Node Package Manager (NPM) 312
- NuGet developer tools and assemblies
- about 136, 139
 - Code generation tool 136
 - configuration migration tool 140
 - package deployer tool 140
 - Plugin registration tool 137
 - solution packager tool 140
- ## O
- OAuth authentication
- about 249, 250
 - types 249
- OData data source 402
- Office 365 Activity Logging
- about 95, 96
 - data, collecting for events 95
- Office 365 Security & Compliance Center 95
- Office 365 authentication 248
- OneDrive for Business

- Dynamics 365 applications,
integrating with 350, 351
- On-Premises Data Gateway
about 28, 73
components 74
types 73
- Open Data Initiative (ODI) 23
- operational acceptance testing (OAT) 106
- Organization Service (SOAP) 136
- P**
- Package Deployer installation 401
- package deployer tool 140
- pass-through authentication approach
about 240
considerations 241
implementing 241
- password hash synchronization approach
about 239
considerations 240
implementing 240
- PCF code components
implementation 311
manifest 311
resources 311
- pipeline 167
- Platform as a Service (PaaS) 51
- plugin registration tool 137
- portal authentication
external authentication 254
local authentication 253
types 254
- Postman
capabilities 137
URL 137
- Power App Portals 348
- Power Apps 384
- Power Apps Command-Line
Interface (CLI) 135
- Power Apps Component Framework
(PCF) 135, 222, 311, 312
- Power Apps portals
authorization 271
capabilities 29
- Power Apps Portals, components
CDS Configuration and metadata 331
Microsoft Azure 331
- Power Apps Portals extensibility 331, 332
- PowerApps Portal Studio
about 124
capabilities 124
- Power Automate
about 26, 307, 308, 384
administering 92
authorization 269
automated flows 26
button flows 26
monitoring 92
scheduled flows 26
UI flows 26
- Power Automate, CDS connectors types
Common Data Service Connector 307
- Power Automate designer
about 126
capabilities 128
- Power Automate flow
creating 27
- Power Automate UI Flows
reference link 367
- Power BI
ALM, using for 168
authorization 270
capabilities 28
capacity 75
components 169

- dashboards 76
- dataflows 75
- datasets 75
- environments 168
- key concepts 28
- reports 76
- structure 75, 76
- workbooks 75
- workspaces 75
- Power BI admin center
 - admin center, URL 84
 - capabilities 84, 85
- Power BI ALM
 - about 169, 170
 - using 178
- Power BI authentication 251
- Power BI Builder
 - about 132
 - capabilities 132
- Power BI dataflows 385
- Power BI deployment pipelines 170
- Power BI designer tools 130
- Power BI Desktop
 - about 130
 - capabilities 131
- Power BI extensibility
 - options 335, 336
 - presenting 335, 336
- Power BI Service
 - about 133
 - URL 132
- Power BI-specific App registration tool
 - URL 251
- Power BI visualizations
 - about 384
- integrating, into Power Platform solutions 384, 385
- Power Platform
 - administration and monitoring 81
 - administration centers 81
 - administration, using Azure DevOps 94
 - API administration 89
 - application monitoring 96
 - architecture 62
 - auditing 94
 - auditing technologies 94
 - authorization 256, 257
 - Azure DevOps for 161
 - clients 76
 - Customer Cloud structure 64
 - data protection standards 273, 274
 - environments 67
 - integrating, with Microsoft 365 347
 - integrating, with Microsoft Azure 347
 - Microsoft Cloud infrastructure 63, 64
 - Power Automate administration 92
 - Power BI admin center 84
 - Power BI's structure 75
 - PowerShell administration 85
 - technology 66, 67
- Power Platform Admin Center
 - administration and monitoring
 - capabilities 83
 - capacity analytics 83
- Common Data Service
 - analytics, areas 84
 - URL 82
- Power Platform Administration Center 160
- Power Platform-based solution
 - ALM, using for 148
- Power Platform build tools
 - Azure DevOps, using with 163
- Power Platform Build Tools
 - about 142

- capabilities 142
- environment tasks 162, 163
- helper tasks 161
- overview 161
- quality check tasks 161
- solution tasks 162
- Power Platform clients
 - desktop clients 76
 - mobile clients 80
- Power Platform Connectors
 - about 72
 - capabilities, for Power Automate flows 72
 - custom connectors 72
 - premium connectors 72
 - standard connectors 72
- Power Platform environment components
 - about 68
 - Common Data Service (CDS) 69
 - Data Loss Prevention (DLP) policies 72
 - On-Premises Data Gateway 73
 - Power Platform Connectors 72
- Power Platform environments
 - about 67, 68
 - components 68
 - default 67
 - developer 67
 - parameters 68
 - production 67
 - sandbox 67
 - solution dependencies 154
 - solution layering 154
 - solution layering, behavior 156
 - solution segmentation 154
 - support 67
 - trial 67
- Power Platform environment,
 - solution layering
 - managed layer 155
 - system layer 155
 - unmanaged layer 155
- Power Platform extensibility
 - best practices 336
- Power Platform hotfixes 160
- Power Platform implementation approach
 - authentication providers 185
 - customer enterprise architecture and environment 183, 184
 - data protection requirements 185
 - data residency requirements 184
 - documentation, testing 200
 - documentation, training 200
 - documents 200
 - internet restrictions 185
 - overview 182
 - programs and projects 186
 - project documentation 229
 - project implementation
 - methodologies and tools 186
 - project plan 229
 - project plan, creating 197, 198
 - project setup 201, 230, 231
 - project setup and methodology 228, 229
 - project tools 229
 - recommendations 183
 - requirements document 198, 199
 - solution architecture document 199
 - solution/technical design document 199
- Power Platform implementation approach, data migration lead 209
- infrastructure consultant 209
- integration lead 209
- iterative design 223
- release manager 209

- Power Platform implementation approach,
 central roles and responsibilities
about 203
customer procurement 203
customer project sponsor 203
executive steering committee 203
partner sales team 203
Project Management Office (PMO) 203
- Power Platform implementation approach,
 customer roles and responsibilities
about 209
adoption and change
 management (ACM) 213
architecture board 212
business process owner 210
customer program manager 209
customer project manager 210
development 212
enterprise architecture (EA) 212
IT lead 211
IT services 211
key user/tester 210
- Power Platform implementation
 approach, operation phase
about 225
decommission 227
operations 226
operations, first-level support 227
operations, maintenance 226, 227
operations, second-level support 227
operations, support 227
operations, third-level support 227
support transition 226
- Power Platform implementation approach,
 partner roles and responsibilities
about 204, 209
business consultant 207
developers 208
- development lead 208
lead architect 205
program manager 204
project manager 205
solution architect 206
technical consultant 207
test team 208
- Power Platform implementation
 approach, preparation phase
approval, finding 216
budget, specifying 215
contract, signing 219
demand, identifying 214
discovery 218
economic feasibility 214
feasibility 214
hackathon 217
legal feasibility 215
negotiations 218
operational feasibility 215
Proof of Concept (POC) 217
Request for Information
 (RFI), issuing 216
RFP/RFQ/RFT, issuing 216
scheduling feasibility 215
technical feasibility 214
- Power Platform implementation
 approach, project
starting 228
- Power Platform implementation
 approach, project documentation
creating 197
- Power Platform implementation
 approach, project effort estimation
about 192, 195
business requirements 192
custom development 193
data migration 194, 195

- infrastructure requirements 193
- integration 193, 194
- Power Platform implementation
 - approach, project execution phase about 219
 - final testing 224
 - initial analysis 220, 221
 - iterative analysis 221, 222
 - iterative design 222
 - iterative development 223
 - iterative execution 221
 - iterative testing 224
 - project initiation 220
 - project preparation 219
 - solution deployment 225
- Power Platform implementation
 - approach, project implementation methodologies about 186, 187
 - agile model 189, 190
 - combined model 191
 - iterative model 190, 191
 - Waterfall model 188
- Power Platform implementation approach, project management tools about 195
 - Azure DevOps 196, 197
 - effort estimators 197
 - Microsoft Project 195
 - Microsoft Project desktop client 196
 - Microsoft Project for the Web 196
 - Microsoft Project Online 196
- Power Platform implementation
 - approach, project phases about 213
 - preparation phase 213
- Power Platform implementation
 - approach, project types about 201
 - external project 201
 - internal project 201
 - project roles and responsibilities 202
- Power Platform integration
 - designing 346
 - overview 346, 347
- Power Platform integration, best practices about 385
 - backend integrations 386, 387
 - frontend integrations 385, 386
- Power Platform integrations 383
- Power Platform mature security model establishing 282, 283, 284
- Power Platform solution
 - architecting 62
 - designing 290
 - features 182
- Power Platform solution
 - complexity 147, 148
- Power Platform solution management
 - environment variables 150
 - managed properties 153
 - overview 149, 150
 - properties 151
 - solution types 152
- Power Platform solution
 - management, properties
 - configuration page 152
 - solution publisher 151
 - Solution version 151
- Power Platform solution
 - management, solution types
 - default solution 153
 - managed solution 152, 153
 - unmanaged solution 152

- Power Platform solutions
 configuration and customization
 tools 119
 tools and techniques 118
 using 177
Power Platform solution security
 designing 234
PowerShell administration
 about 85
 automation capabilities,
 monitoring 88, 89
 Microsoft 365 administration 85, 86
 Power Apps administration 86, 87
 Power Apps administration, modules 86
 Power BI administration 87, 88
 Power BI administration,
 modules 87, 88
 URL 85
Power Virtual Agents
 about 383
 capabilities 27
 creating 27
 supported standard integration
 channels 383
Power Virtual Agents designer
 about 128
 URL 128
privacy 273
processes 303
production environments
 about 106
 developer sandbox environment 109
 feature testing environment 109
 multiple release strategy 107
 product, upgrading 108
 simple production 106
 training environment 109
- program manager
 activities 204, 205
program managers (PrgMgrs) 231
project initiation
 tasks 220
Project Management Office (PMO) 203
Project Manager (PjMgrs) 231
project preparation
 tasks 219
Proseware Inc. 228
Publish Pipeline Artifact 167
publish-subscribe pattern
 374, 375, 376, 377
- ## R
- relay pattern 371-373
release 167
remote procedure call (RPC)
 pattern 368-370
reports 366
request-callback pattern 377, 379
Request For Proposal (RFP) 216
Request For Quotation (RFQ) 216
Request For Tender (RFT) 216
resources 311
Return Of Investment (ROI) 214
ribbon 123
Ribbon Workbench 123
robotic process automations (RPA) 367
role-based access control (RBAC) 236
row-level security (RLS) 270
- ## S
- scale groups 66
scope levels, DLP policies
 environment level 73

- tenant level 73
- seamless single sign-on 240, 246
- security best practices
 - CDS content-based security 276
 - CDS security roles 274
 - identity and access management
 - automation, using 281, 282
 - integration, across solution
 - components 278
 - Power Platform mature security
 - model, establishing 282-284
 - preparing 274
 - security, integrating across
 - solution components
 - about 278, 279
 - CDS-Power BI integrated security 280
 - CDS-SharePoint integrated security 279
 - standard integration approach,
 - working 278
 - service authentication, for internal users
 - about 247
 - Common Data Service
 - authentication 247
 - Power BI authentication 251
 - service protection API Limits
 - reference link 415
 - several related entities 398
 - SharePoint
 - Dynamics 365 applications,
 - integrating with 350
 - single sign-on options. *See also* seamless single sign-on
 - Site Map 24, 122
 - Skype/Skype for Business
 - Dynamics 365 applications,
 - integrating with 354, 355
 - Software as a Service (SaaS) 54, 67, 246
 - solution architect
 - responsibilities 206
 - solution dependencies 157
 - solution deployment
 - activities 225
 - Solution Layers 156
 - solution packager tool 140
 - solution patch 158
 - solution publisher 151
 - solution segmentation 154, 157
 - solutions management 149
 - solution tasks 162
 - solution type
 - default solution 153
 - solution updates 158, 159
 - solution version 151
 - SQL Azure
 - integrating with 358, 359
 - SQL Server Integration Services
 - data, extracting 403-406
 - data, loading 406
 - data, transforming 406
 - using 403
 - SQL Server Integration Services (SSIS) 321
 - SQL Server Management Studio (SSMS) 321
 - SQL Server Reporting Services (SSRS) 277
 - standard Azure data integrations 381
 - standard role-based security 263, 264
 - subareas 122
 - support transition
 - activities 226
 - System Center Configuration Manager 143
 - system integration testing (SIT) 106

T

Tabular Data Stream (TDS) 248, 321, 387
technical consultant
 development lead 208
 responsibilities 207
tenant 64
testing tools 138
test team
 development lead 208
Transparent Data Encryption (TDE) 274

U

UI flows
 about 270
 using 367
Unicode data encoding 410
Unified Service Desk extensibility
 about 332
 integrate 333
Unified Service Desk (USD)
 about 332, 367
 using 368
Unified Service Desk (USD), components
 USD client 332
 USD packages 332
Unified Service Desk (USD), possibilities
 configure 332
 develop 332
 integrate 333
 integrate telephony 332
unmanaged solution 152, 153
unsupported integration topology
 about 99
 central consolidation environment 100
 central reporting environment 101
URL 99

USD, desktop clients

Agent Desktop application 78, 79

Common Data Service solution 78

components 78

Omnichannel for Dynamics 365

 Customer Service 79

Robotic Process Automation,
 with UI Flows 79

user acceptance testing

 (UAT) 106, 195, 224

user group, Contoso Inc. Power
 Platform architecture

Power Platform solution users 115

project core team 115

project extended team 115

User Interface Integration (UII) 78

user interface (UI)

 testing 138

V**Views**

 managed properties 153

virtual entities

 using 380, 381

Visual Studio

 about 134

 Power Platform extensions,

 using for 136

 Visual Studio Code 135

W**Waterfall model**

 about 188

 features 188

Web Hooks integration 357

web resources, CDS client-side extensibility
overview 313

X

XrmToolBox
about 123, 137
URL 123