

Microsoft Cloud Developer Docs

Learn how to integrate services across the Microsoft Cloud.

Azure and Microsoft 365 Tutorials



[Audio/Video Calling from a Custom App into a Teams Meeting](#)

[Build productivity apps by using Microsoft Graph Toolkit](#)

[Integrate OpenAI, Communication, and Organizational Data Features into a Line of Business App](#)

Azure and Power Platform Tutorials



[Automate Data Reporting with Azure Functions and Power Automate ↗](#)

[Java + Power Apps Workshop ↗](#)

[Power BI and Azure Synapse Workshop using SerengetiDataLab ↗](#)

[Transform your Business Applications with Fusion Development](#)

Build Applications on the Microsoft Cloud



[Introduction](#)

[Create and deploy apps in less time](#)

[Get the most value from technical talent](#)

[Integrate apps with existing solutions](#)

[Create and run secure apps](#)

[Summary](#)

Copilot

 **GET STARTED**

[Adopt, extend and build Copilot experiences](#)

[Copilot for Dynamics 365](#)

Build resilient apps with Dev Proxy

 **GET STARTED**

[What is Dev Proxy?](#)

[Get started](#)

[Tutorials](#)

[How-to](#)

[Reference](#)

[Concepts](#)

Build applications on the Microsoft Cloud

Article • 12/05/2023

ⓘ Note

This is article 1 of 6 in **Build applications on the Microsoft Cloud**.

The Microsoft Cloud provides a unified collection of services for creating applications. Enterprise application development leaders can combine these services to deliver more value in less time with better security. This guide describes the big picture of Microsoft Cloud from an application development perspective, focusing on what leaders must do to be successful.

The Microsoft Cloud: An application development perspective

Most new applications use cloud services in some way. Many organizations rely on Microsoft Azure and the services it provides for professional software developers. This is fine, but you can create better applications by using other Azure services also.

To see this, start by recognizing that Azure is just one part of the Microsoft Cloud. From an application development point of view, the Microsoft Cloud also includes several other components, as Figure 1 shows. Your organization probably uses some or all of these today. You might not think of them as part of your application platform, but they are. Each one offers useful cloud services for your applications to use. So why not exploit them to make your new applications the best they can be?

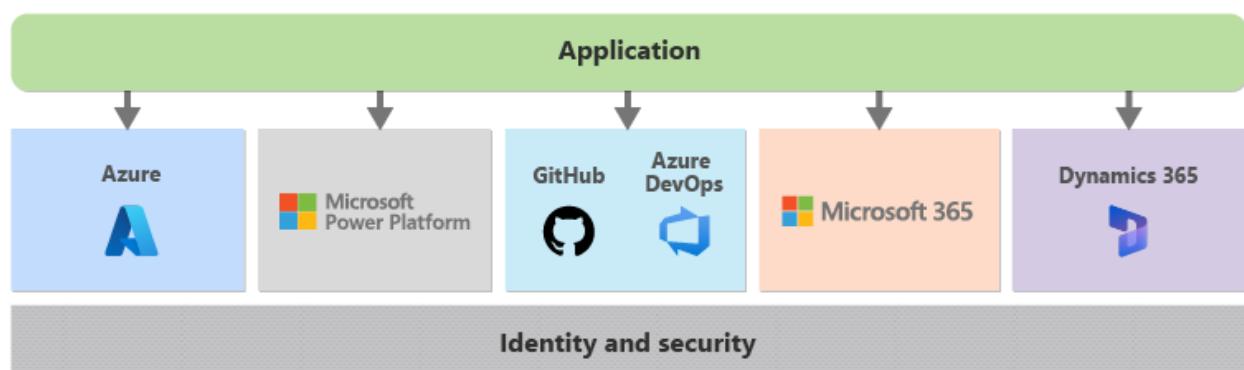


Figure 1: The Microsoft Cloud provides a unified set of services that applications can use.

These services work together and are an integrated platform for application development.

- [Microsoft Azure](#) has services aimed at professional software developers who create and maintain new applications.
- [Power Platform](#) has services for low-code development. Citizen developers—people who aren't software professionals—can use the services to create applications and automate business processes.
- [GitHub](#) and [Azure DevOps](#) have tools to create, test, deploy, and run applications.
- [Microsoft 365](#) has applications for the modern workplace, including Microsoft Teams, Exchange, and SharePoint.
- [Dynamics 365](#) has applications for business, including applications for sales support, financials, and customer service.
- [Microsoft Entra ID](#) and [Microsoft Sentinel](#) are foundational services for securing applications and systems.

Building new solutions on this kind of broad, unified cloud platform is the next step in the evolution of application development. By providing an integrated set of widely used services, the Microsoft Cloud offers a foundation for doing this.

Using these unified services brings real benefits, including:

- You can easily combine different technologies. For example:
 - Combine a low-code application with more complex software created by professional developers.
 - Build a low-code application that's integrated with a collaboration platform.
- You can integrate and share data more easily. For instance, you can have a new cloud-native application access data from line-of-business applications or modern work solutions.
- There's less integration risk, because the Microsoft cloud services work well together.
- There's better security, because using the same security services and tools across the entire environment is simpler for application developers. It also makes life easier for the people who monitor and manage applications in production.

How enterprise application development leaders succeed

To succeed in a modern enterprise, an application development leader needs to do several things. To help you understand how to succeed at them, the articles in this

This section walk you through a simple scenario of a typical enterprise application. Along the way, you'll see examples of the value that each Microsoft Cloud component brings.

The following table describes each article and provides a link to it. The articles are meant to be read in order.

[\[+\] Expand table](#)

How enterprise application development leaders succeed	Article that shows how
They create better applications in less time with pro-code development on Azure and low-code development on Power Platform.	2. Create and deploy more applications in less time
They get the most value from technical talent with fusion development and an integrated low-code and pro-code development process that includes GitHub and Azure DevOps.	3. Get the most value from technical talent
They integrate new applications with existing solutions by using Azure API Management, Microsoft Graph, and Dynamics 365.	4. Integrate new applications with existing solutions
They create and run secure applications by using Active Directory for identity and access management. They also use Azure Sentinel for security information and event management (SIEM), and for security orchestration, automation, and response (SOAR).	5. Create and run secure applications

After you read these articles, you can learn more about building applications on the Microsoft Cloud by making use of our [Microsoft Cloud resources collection](#). It provides design guides, solutions, learning resources, and product documentation to help get you started.

Next steps

See how successful enterprise application development leaders create better applications in less time with pro-code development on Azure and low-code development on Power Platform.

[2. Create and deploy more applications in less time](#)

Create and deploy more applications in less time

Article • 12/05/2023

ⓘ Note

This is article 2 of 6 in **Build applications on the Microsoft Cloud**.

Most enterprise application development leaders share a goal: to create and deploy better applications in less time. This article shows how you can use the Microsoft Cloud to reach this goal.

- Use Microsoft Azure to succeed with pro-code development
- Use Power Platform to succeed with low-code development

Use Microsoft Azure to succeed with pro-code development

Professional software developers are the heart of enterprise development organizations. These pro-code developers create custom software using C#, Java, Python, and other programming languages. They also use powerful database systems, messaging services, and other technologies aimed at professional developers.

To support pro-code developers, the Microsoft Cloud provides Microsoft Azure, which has a broad set of services. Your organization can choose the best options for the applications that you need to create.

- [Azure Virtual Machines](#), for deploying Linux and Windows virtual machines.
- [Azure App Service](#), an HTTP-based technology for running web applications and mobile back ends. The software can be written in many different languages, and can run on either Windows or Linux.
- [Azure Kubernetes Service \(AKS\)](#), for deploying a managed Kubernetes cluster to run containerized applications.
- [Azure Functions](#), for creating callable blocks of code, called functions, then automatically scaling as needed to handle client requests. This approach is an example of serverless computing.
- [Azure Static Web Apps](#), for automatically building and deploying web applications to Azure, triggered by changes made to application source code in GitHub or in Azure DevOps repositories. Static Web Apps can also host serverless application

APIs created with Azure Functions, and deploy front-end web applications built with Angular, React, and other frameworks.

- [Azure Logic Apps](#), a service that makes it possible for developers to create and run automated workflows. These workflows can integrate applications and data, so that your organization can quickly develop scalable integration solutions for enterprise and business-to-business (B2B) scenarios.
- Relational data services, including:
 - [Azure SQL Database](#)
 - [Azure Database for MySQL](#)
 - [Azure Database for PostgreSQL](#)
- [Azure Cosmos DB](#), a fully managed NoSQL database with scalable support for several different approaches to working with data.

To get a sense of how pro-code developers can use Azure, suppose that an organization needs to create a custom application for use both by customers and employees. For example:

- A healthcare organization wants to provide a way for patients to access test results that medical professionals upload.
- Customers of a financial services firm apply for loans that must be approved by the firm's employees.

In scenarios like these, it's likely that the application has distinct components that interact with customers and employees and that share data. Figure 2 shows the basics of this solution.

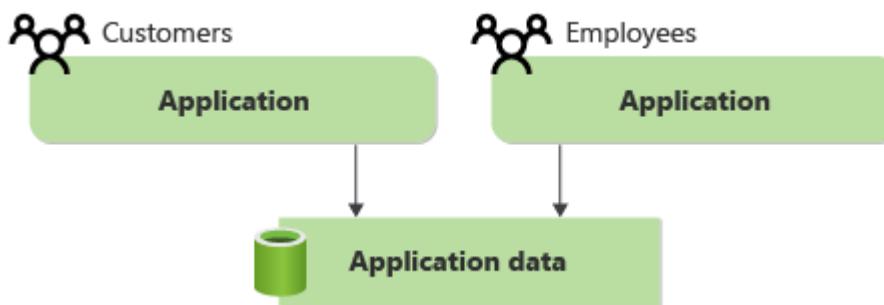


Figure 2: In many modern enterprise applications, customers and employees work with shared data.

To keep the application responsive when it serves a large number of simultaneous users, the pro-code developers can build the customer-facing software on Azure. They might choose to use a microservices architecture on containers that run on Azure Kubernetes Service, or perhaps a simpler approach using a web application hosted in Azure App Service. Either way, your developers will also need to choose a data service for the application to use. Figure 3 shows how this looks.

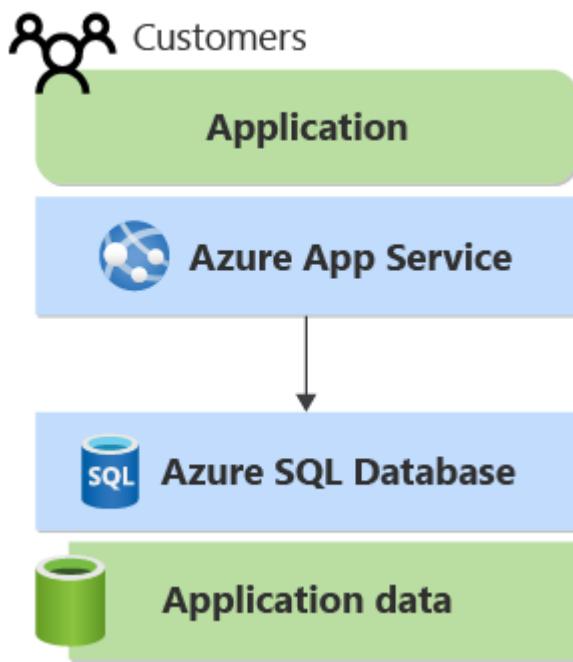


Figure 3: Pro-code developers can use Azure App Service and Azure SQL Database to create the customer-facing part of the application.

In our example, the development team chooses to create the customer-facing application by using Azure App Service and Azure SQL Database. The result is a scalable, reliable application that works effectively with the organization's customers.

Besides providing application development tools, Microsoft Cloud also provides operations tools:

- [Azure Monitor](#) is a service that monitors applications to maximize their availability and performance. It collects metrics that describe various aspects of a system, and it creates logs that contain events, traces, and performance data.
- [Application Insights](#) is a feature of Azure Monitor for managing application performance.
- [Log Analytics](#) is a tool in the Azure portal for querying log data.

The [Azure Well-Architected Framework](#) has guidance to help professional developers create better Azure applications. The [Cloud Adoption Framework](#) has guidance and best practices for adopting and governing Azure.

Use Power Platform to succeed with low-code development

You can create some applications quickly and easily with low-code tools, which are tools that can be used by people who aren't software professionals. Whenever possible create applications the low-code way, not the pro-code way, to create more applications in less

time, and to save money. Low-code development can be done either by software professionals or by citizen developers who aren't software professionals.

The Microsoft Cloud provides Power Platform for low-code development. It includes these services:

- [Power Apps](#) for building low-code applications
- [Power Automate](#) for creating flows to automate business processes
- [Power Virtual Agents](#) for building chatbots
- [Power BI](#) for creating data-driven insights

Although all of these services can be used by non-professional developers, low-code development can also have real value for professional developers. Often they can use Power Platform to create an application more quickly than if they used a language like C#.

In our example application, we create the employee-facing component by using Power Apps. Figure 4 shows how this looks.

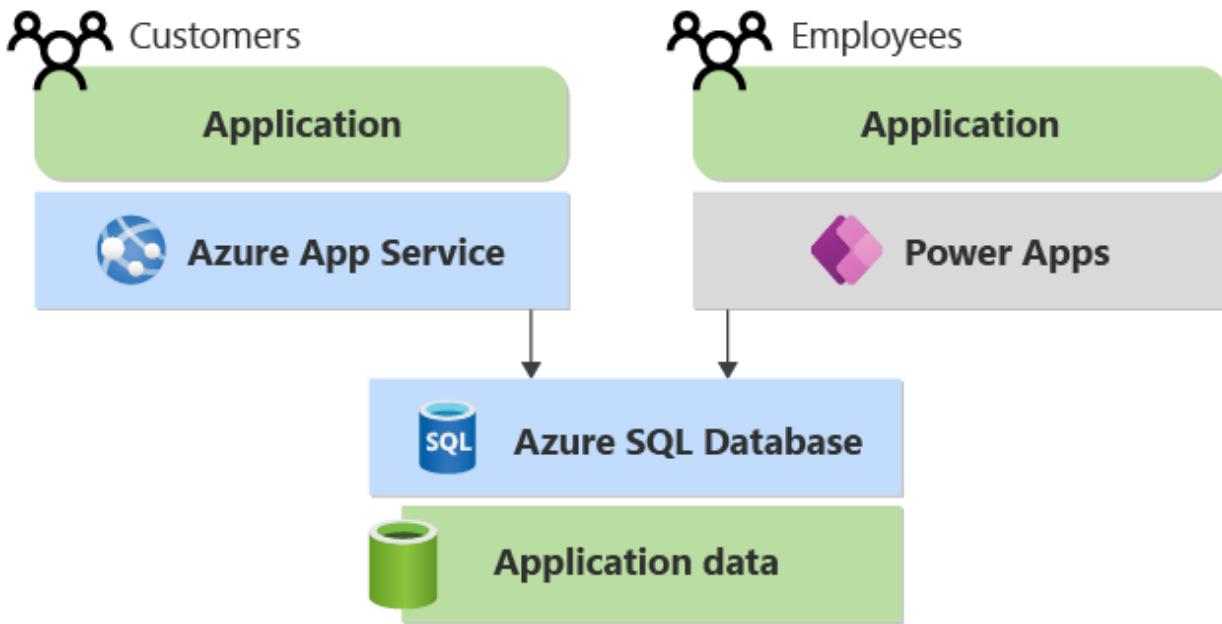


Figure 4: Citizen developers or professional developers can use Power Apps to create a low-code application for the employee-facing part of the solution.

Here are benefits of the low-code approach:

- A low-code application can be ready in less time. Power Apps developers commonly use Power Apps Studio to create an application quickly with a point-and-click approach—they don't need to write code. If the application is created by citizen developers, they don't have to wait for professional developers to be available. They can create and update the application on their own schedule.

- A low-code application can easily connect to many kinds of data. In this example, the low-code application uses Azure SQL Database for its data, just like the pro-code part of the solution. Low-code applications can also work with many other data stores, such as Dataverse—designed for use with Power Platform—or Azure Cosmos DB. An application accesses a data store by using a connector. A developer can quickly add a connector to an application by dragging and dropping the connector into the application.
- Connectors make it possible for a low-code application to work with many other technologies. For example, Microsoft provides connectors to applications and data from many sources, such as Oracle, Salesforce, Dropbox, and SAP. There are more than 450 connectors. An application can even use connectors to access functionality that's provided by other cloud services, such as sending tweets with Twitter.

Low-code development is a technology that gives your organization more ways to build applications and to build them quickly. It's an essential way to create better applications in less time.

When appropriate, professional and citizen developers can work together to create an application. This approach, called fusion development, is discussed in a companion article, [3. Get the most value from technical talent](#).

Deploying applications with Microsoft Teams

However they're built, your applications have value only if they're used. For this reason, it's a good idea to embed applications in an environment where the users are.

An important example of this is deploying applications within Microsoft Teams. Pro-code applications created by using Azure can use this option, as can low-code applications that are built on Power Platform. Teams is used by hundreds of millions of people every month, so why not connect your applications to this popular technology?

For low-code applications, there's another powerful option: using [Dataverse for Teams](#). It's part of Microsoft Teams, which is part of Microsoft 365, so typically there's no extra license required to use it. Dataverse for Teams also includes subsets of Power Automate, Power Virtual Agents, and Power BI.

Here are some benefits of using Dataverse for Teams:

- It simplifies the deployment of applications and data to team members.

- You can control the permissions for the application by using Teams. Rather than working with the more detailed options available with Power Apps itself, you can use the permissions already established for the members of a team. This is simpler and less likely to lead to mistakes.
- You get a uniform user interface style for your low-code applications. Rather than letting each group of citizen developers define their own approach, they can match the style of Teams to make the applications easier to understand and use.

Whether or not you use Dataverse for Teams, deploying applications within Teams is a clear example of the value provided by the unified services of the Microsoft Cloud.

Next steps

See how successful enterprise application development leaders get the most value from technical talent with fusion development and an integrated low-code and pro-code development process that includes GitHub and Azure DevOps.

3. Get the most value from technical talent

Get the most value from technical talent

Article • 12/05/2023

ⓘ Note

This is article 3 of 6 in **Build applications on the Microsoft Cloud**.

Using both pro-code and low-code development can help your organization get the most from on-staff talent. It also changes the way you create software. This article shows ways to use the Microsoft Cloud to get the most from your talent.

- Use Power Platform and Azure for fusion development
- Use GitHub and Azure DevOps to create an integrated development process

Use Power Platform and Azure for fusion development

Low-code development is useful, but it's not the best solution for everything. In our example application, for instance, the employee-facing part was built on Power Platform, but because the customer-facing part required greater scalability and better performance, it was built by pro-code developers on Azure.

Creating complete solutions that combine low-code and pro-code is known as fusion development. Being successful with fusion development requires making good decisions about when to use low-code and when to use pro-code. In general, you should consider using low-code whenever possible, especially for employee-facing applications, because it's likely to be less expensive and take less time to build.

You shouldn't expect to do everything with low-code. A pro-code approach is better for apps that must handle tens of thousands of simultaneous users with good response. One way to meet these requirements is to use a microservices architecture and run on Kubernetes, but this can't be done by using Power Platform.

With Azure and Power Platform, the Microsoft Cloud provides an integrated approach to fusion development. Here are some benefits of this approach:

- Power Platform is built on Azure, so they work well together. For example, applications created on either foundation can easily use the same data sources, as shown in [2. Create and deploy more applications in less time](#). Both also use the same underlying technologies for identity and security.

- Power Platform makes it easy for pro developers to create useful extensions for low-code development. For example, pro developers can use Visual Studio to create custom controls for low-code user interfaces. They can also create custom connectors for accessing applications and data.
- Developers using both Power Platform and Azure can rely on the same technologies for managing source code and deploying applications. The next section looks at this approach in more detail.

Low-code and pro-code development are both important for your organization, and successful enterprise development groups typically do both. They embrace fusion development.

Use GitHub and Azure DevOps to create an integrated development process

Although both professional and non-professional developers can use Power Apps to create useful applications, it's still important to use a modern development process. Professional developers can help citizen developers understand and use a more professional approach to creating and deploying software. The Microsoft Cloud provides two technologies to implement professional development processes:

- [GitHub](#) provides source code control and other services that help developers collaborate on the same code base. By using GitHub Actions, developers can create workflows that automatically build and deploy code.
- [Azure DevOps](#) also provides source code control and other services that help developers collaborate. Developers can use DevOps to create pipelines, which are like workflows, that build and deploy code.

GitHub and Azure DevOps overlap in some ways. Both can help developers automate build and deployment, for example. They can also be used together. For example, you can use GitHub for source control and Azure DevOps to implement a pipeline.

Both GitHub and Azure DevOps were created to meet the needs of professional developers, and both are widely used today for this purpose. But both technologies are part of the Microsoft Cloud and can be used by low-code developers to create apps for Power Platform. This means that your organization can use GitHub, Azure DevOps, or both, to support fusion development teams. Figure 5 illustrates this idea.

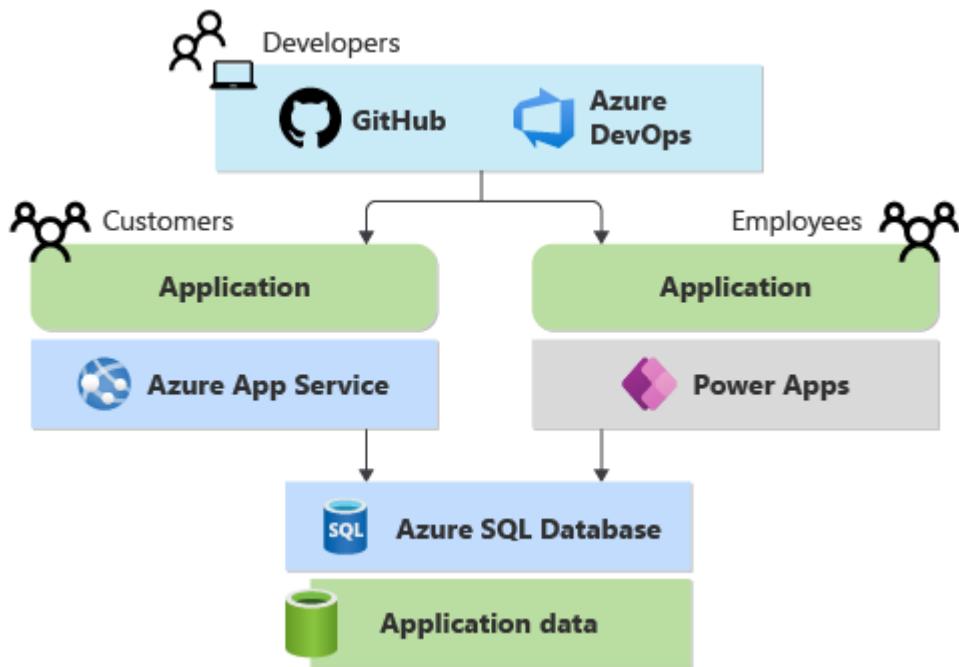


Figure 5: Pro-code and low-code development can use an integrated development process.

Here are some benefits of using GitHub and Azure DevOps to support fusion development:

- Just as a group of pro-code developers working together to create a C# application can use GitHub and Azure DevOps for source code control, a group of low-code developers working together to create a Power Apps application can take advantage of these same technologies. Much like pro developers, Power Apps developers can extract apps from a repo, modify them, then return the updated version. This integration gives citizen developers the ability to use a more disciplined development process, improving the quality of the apps they create.
- Just as pro-code developers automate deployment with GitHub workflows and Azure Pipelines, Power Platform developers can use these same technologies to deploy a solution into the correct environment. This pipeline is typically simpler than one used by pro-code developers. Power Platform solutions aren't compiled, for example, so there's no build step.

Using the Microsoft Cloud makes possible a modern process for fusion development that's built on shared technologies including GitHub and Azure DevOps.

Use new technologies for business innovation

One of the best ways to improve your organization is to take advantage of new technologies that facilitate business innovation. Exploiting these advances can lower

your costs, reach more customers, and help you enter new lines of business. In fact, ignoring new technology is dangerous: why let your competitors get these benefits first?

The Microsoft Cloud provides an ongoing stream of leading-edge technologies for your organization. For example, one important area for innovation is artificial intelligence (AI) and machine learning. Among the advanced AI technologies in the Microsoft Cloud are:

- [Azure AI Services](#), which makes it possible for professional developers to exploit machine learning models from Microsoft in their applications. The services available include Computer Vision for processing images, an API for facial recognition, a Language service for understanding natural languages, and a speech service that makes it possible for your applications to have capabilities such as speech-to-text and text-to-speech conversion, and speech translation.
- [AI Builder](#), a Power Platform capability that makes it possible for low-code developers to create and use their own custom machine learning models. It also provides prebuilt models for common business scenarios such as object detection and extraction of information from invoices or other business documents.
- [Power Apps Ideas](#), which gives low-code developers the ability to specify application behavior by expressing what they want in plain natural language. These requests are automatically translated into the Power Fx formulas that Power Apps uses. This translation relies on OpenAI GPT-3, one of the most advanced natural language models in the world.

Another important area for innovation is the Internet of Things (IoT)—networks of physical devices that exchange data with one another and with services. There are billions of connected devices in the world, and more to come. To help your organization take advantage of IoT, the Microsoft Cloud offers such technologies as:

- [Azure IoT Hub](#), a managed service hosted in the Microsoft Cloud that acts as a central message hub for communication between an IoT application and its attached devices. Your organization can use this service to connect millions of devices to back-end solutions reliably and securely.
- [Azure IoT Edge](#), which extends IoT Hub by analyzing device data on-premises by using edge applications instead of cloud applications. Edge applications react to events more quickly than cloud applications, and reduce traffic to the cloud.

A third area for innovation is quantum computing. To help your organization start using this technology, the Microsoft Cloud provides [Azure Quantum](#), with two main paths:

- [Quantum Computing](#), which helps you experiment with and create prototypes using a variety of quantum hardware providers.
- [Optimization](#), focused on using quantum computing to find the best solution from a set of possible options.

Microsoft also provides the Quantum Development Kit, a set of tools, including the open-source Q# language, for creating quantum software.

The Microsoft Cloud provides leading-edge technologies that can help your organization create real business innovation. Keeping abreast of what new technologies can do is an essential part of succeeding as an application development leader.

Next steps

See how successful enterprise application development leaders integrate new applications with existing solutions by using Azure API Management, Microsoft Graph, and Dynamics 365.

[4. Integrate new applications with existing solutions](#)

Integrate new applications with existing solutions

Article • 12/05/2023

ⓘ Note

This is article 4 of 6 in **Build applications on the Microsoft Cloud**.

It's rare that new applications don't connect to existing applications and data. High-quality apps that people want to use typically interact with what already exists. In this article, we consider various ways that you can use Microsoft Cloud to connect to applications and data.

- [Use Azure API Management to connect to your existing applications and data](#)
- [Use Microsoft 365 and Microsoft Graph to connect to your modern work applications and data](#)
- [Use Dynamics 365 to connect to your line-of-business solutions and data](#)

Use Azure API Management to connect to your existing applications and data

[Azure API Management](#) provides a simple, general, and manageable way to connect to applications and data. You can use it to create a standard interface that provides a single point of access for other applications to use. API Management also makes it possible to manage and secure your application APIs in a standard way, and gives developers a single place to learn how to use those APIs.

In our example application, suppose that both the pro-code customer-facing component and the low-code employee-facing component need to access existing applications and data. Figure 6 shows how API Management fits into the picture.

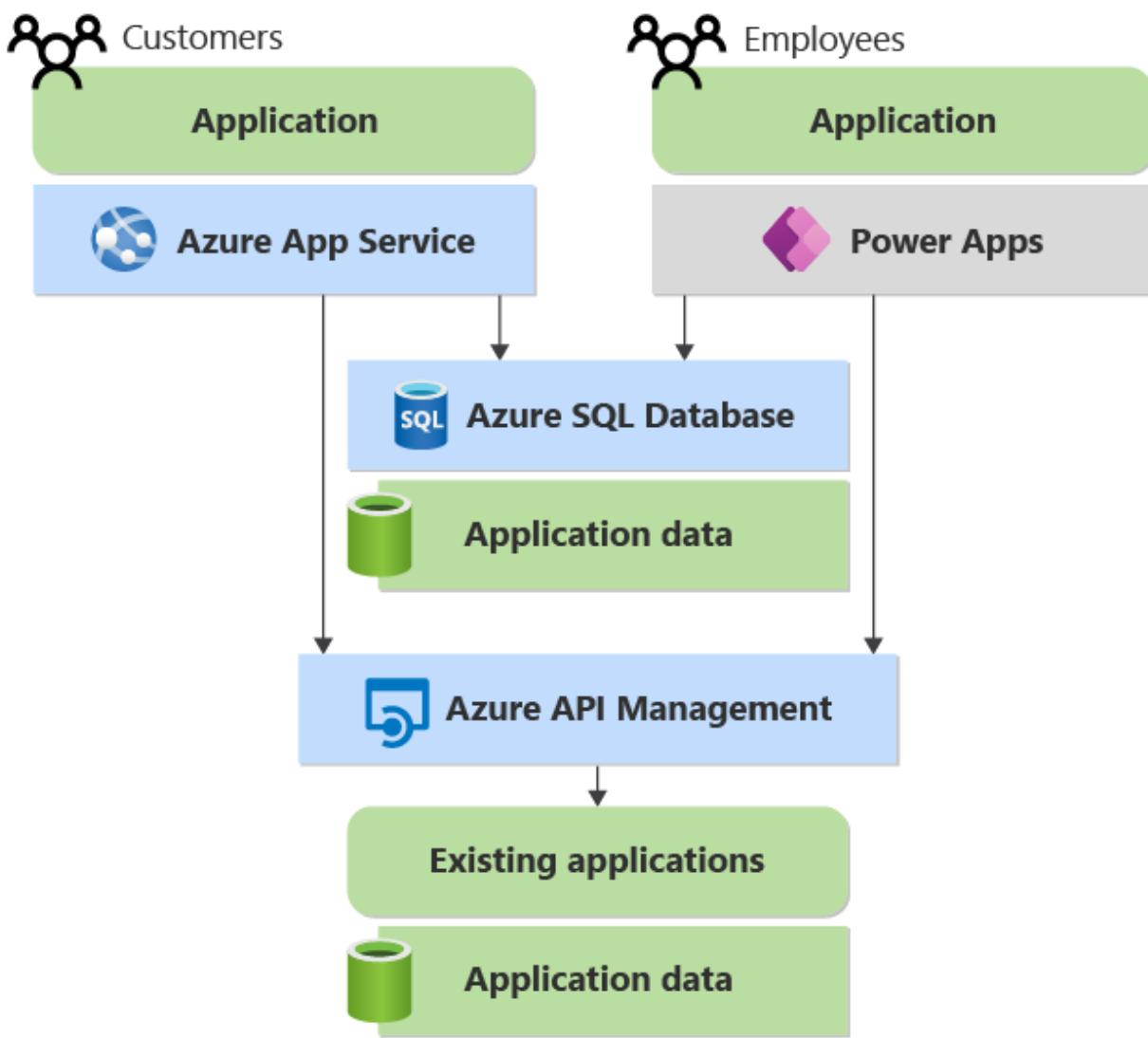


Figure 6: Azure API Management provides a uniform way to wrap existing applications and data.

These existing applications can be running on Azure, on another vendor's cloud platform, in your own datacenter, or elsewhere. It doesn't matter whether the applications were created by Microsoft, by your internal developers, or by another company. Wrapping them with API Management hides these differences so that applications access other applications and data in a standard way.

The benefits of using API Management include:

- Both pro-code and low-code applications can call the APIs directly. In Figure 6, for example, the customer-facing component and the employee-facing component both access existing applications and data through API Management.
- Power Platform applications can use a standard connector to connect to API Management. Like other connectors, this one can be added to a low-code app, providing a simple and consistent way to connect to existing applications.
- Pro-code developers can create custom API wrappers for existing applications and then publish them for use by low-code applications through the API Management

connector. Visual Studio provides a dialog that makes this easy to do.

Connecting new applications to your existing applications and data is important. The Microsoft Cloud provides Azure API Management to solve this problem effectively.

Microsoft Cloud in a hybrid world

Cloud computing is a mainstream approach today, but many organizations still have a substantial investment in on-premises technology. Creating successful applications by using the Microsoft Cloud often requires integrating with this existing investment.

Microsoft Cloud components can connect to your on-premises world in a variety of ways. Here are some examples:

- **Azure:** Many aspects of Azure can connect to your on-premises environment. You can use [VPN Gateway](#) to connect your on-premises network to Azure. [Azure DevOps pipelines](#) can deploy code to the cloud or on-premises. You can run a subset of Azure services on-premises by using [Azure Stack Hub](#), [Azure Stack HCI](#), or [Azure Stack Edge](#).
- **Power Platform:** Microsoft provides an [on-premises gateway](#) so that your low-code apps can access data stored in your datacenter. For example, an application created with Power Apps can use the gateway to access on-premises SQL Server data, and a Power BI solution can use it to combine data from on-premises data sources with data stored in the cloud.
- **Microsoft 365:** Various Microsoft 365 components, including Exchange and SharePoint, can provide hybrid deployment by using connections between cloud and on-premises software.
- **Dynamics 365:** Some Dynamics 365 components, such as Supply Chain Management, allow for a hybrid deployment.
- **Microsoft Entra ID :** It's common today to connect this cloud-based identity service with an existing on-premises Active Directory, creating a hybrid solution for managing identity. Identity information can be automatically synchronized between the two services.

Your organization can also use [Microsoft Sentinel](#) and various aspects of [Azure Arc](#) to secure and manage your infrastructure on Azure, on other public clouds, or in your own datacenter. This ensures that applications you deploy in a hybrid fashion can still be effectively managed.

Use Microsoft 365 and Microsoft Graph to connect to your modern work applications and data

High-quality internal applications help employees do their jobs well. Your pro-code and low-code applications can be more useful if they give employees direct access to their organizational data, such as emails, calendar entries, and spreadsheets. This data is often stored in Microsoft 365 tools such as Exchange, SharePoint, and OneDrive.

For example, a custom application can make its users more productive by providing built-in access to calendar data, emails, and chats. This minimizes the time users spend switching between the application and Outlook or Teams. Applications can also show employees relevant documents from OneDrive to help them make better decisions. They get the data that they need without having to search.

To access this kind of data, your custom applications can use the unified API provided by [Microsoft Graph](#). Figure 7 illustrates this idea in our sample application.

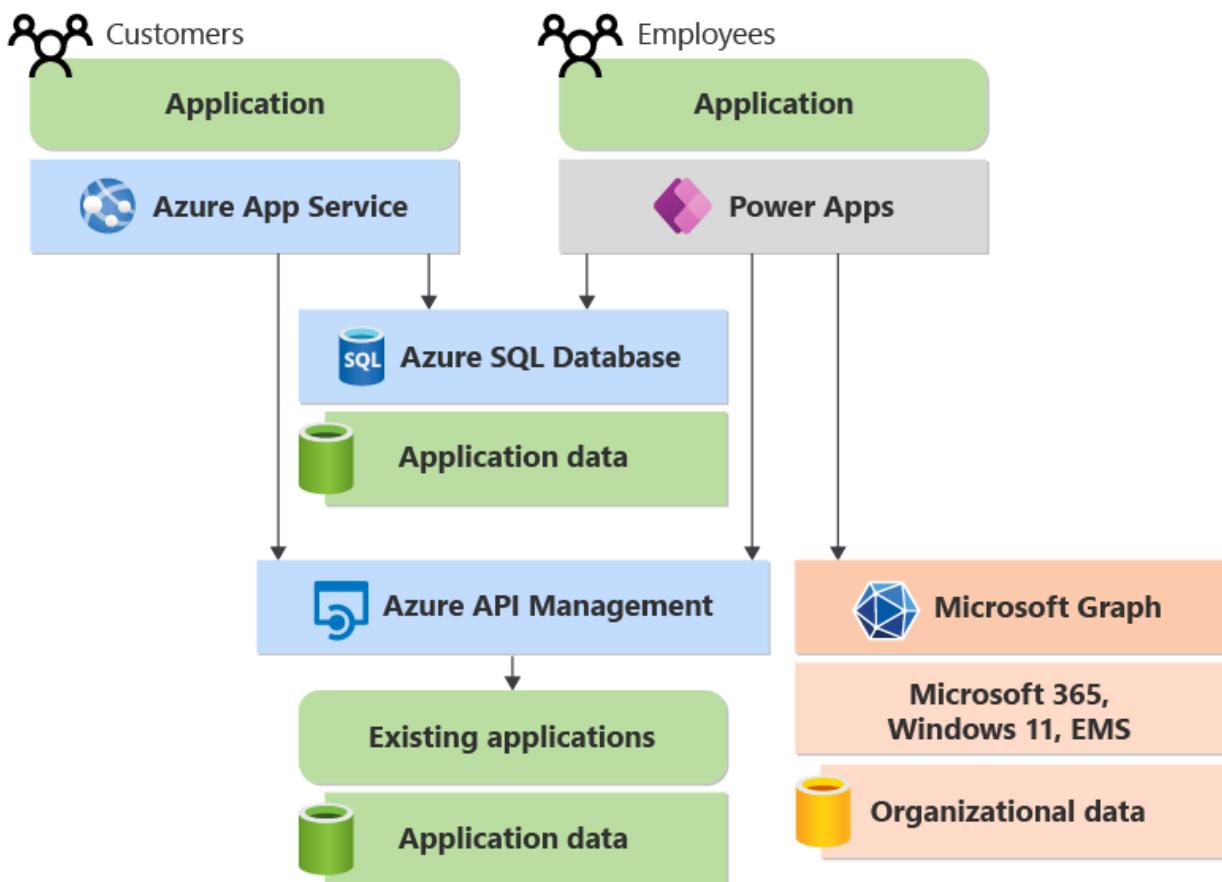


Figure 7: Microsoft Graph gives applications access to the organizational data in Microsoft 365.

Applications can use Microsoft Graph to access data in Microsoft 365, Windows 11, and Enterprise Mobility + Security. The data that this API exposes can come from many different cloud services, including Exchange, SharePoint, Teams, OneDrive, OneNote, Planner, Microsoft Search, and Microsoft Entra ID. Because Microsoft 365 already meets compliance requirements for e-discovery and records management, building applications that use its content and services gives you solutions that are more compliant. Microsoft Graph can be used both by pro-code and low-code apps.

In our example application, customers and employees work together in some way. Suppose that employees exchange mail or engage in Teams chats with these customers. If an employee needs to look up information from a prior mail exchange or chat with a customer, the application can use Microsoft Graph to get it from Microsoft 365. By selecting a customer in the employee-facing application interface, the employee can view the relevant mail exchange or chat.

Applications can use the data accessed via Microsoft Graph in many other ways. For example:

- A custom application can personalize interactions with employees by displaying employee names and pictures rather than just names.
- A custom application can check an internal user's calendar, then automatically suggest times for a meeting with customers. The application can subscribe to changes in the user's calendar and advise of a need to adjust scheduled meetings.
- A custom application can automate employee onboarding. It can automatically add a new employee to appropriate Teams channels, email lists, and other areas in Microsoft 365.
- A custom application that needs to store data can use SharePoint instead of an Azure data service. The application can then use SharePoint features such as permission management and improved compliance.

Microsoft Graph can connect to external data sources including Google Drive, Box, Jira, and Salesforce, so that your custom applications can index and search information stored outside of your Microsoft environment. Microsoft also provides Microsoft Graph Data Connect for moving large amounts of organizational data into Azure datastores. You can then use the data to create machine learning models and do other analysis, helping you better understand what's happening in your organization.

Microsoft Graph is yet another example of the value provided by building applications on the unified services of Microsoft Cloud. You can use it to create better applications that make employees more productive.

Use Dynamics 365 to connect to your line-of-business solutions and data

Dynamics 365 is a set of services that can provide a range of business solutions. Here's a partial list of what it includes:

- [Dynamics 365 Sales](#) for managing sales pipelines, onboarding new sellers, and supporting sales professionals in other ways.
- [Dynamics 365 Customer Insights](#) to help you understand your customers better. For example, it can provide a view of each customer and predict customer needs.
- [Dynamics 365 Supply Chain Management](#) for building and running a resilient supply chain that optimizes your inventory and improves demand planning.
- [Dynamics 365 Customer Service](#) to support your customer service organization. For example, it can get your representatives faster answers and create virtual agents (chatbots).
- [Dynamics 365 Finance](#) to handle invoices, payments, and other financial matters.
- [Dynamics 365 Human Resources](#) to help manage recruitment, employee benefits, compensation, and other HR issues.

New enterprise applications often can benefit from integrating with one or more Dynamics 365 services. Because we use Power Apps to create our example application, we can integrate easily by using a connector. Figure 8 shows how this looks.

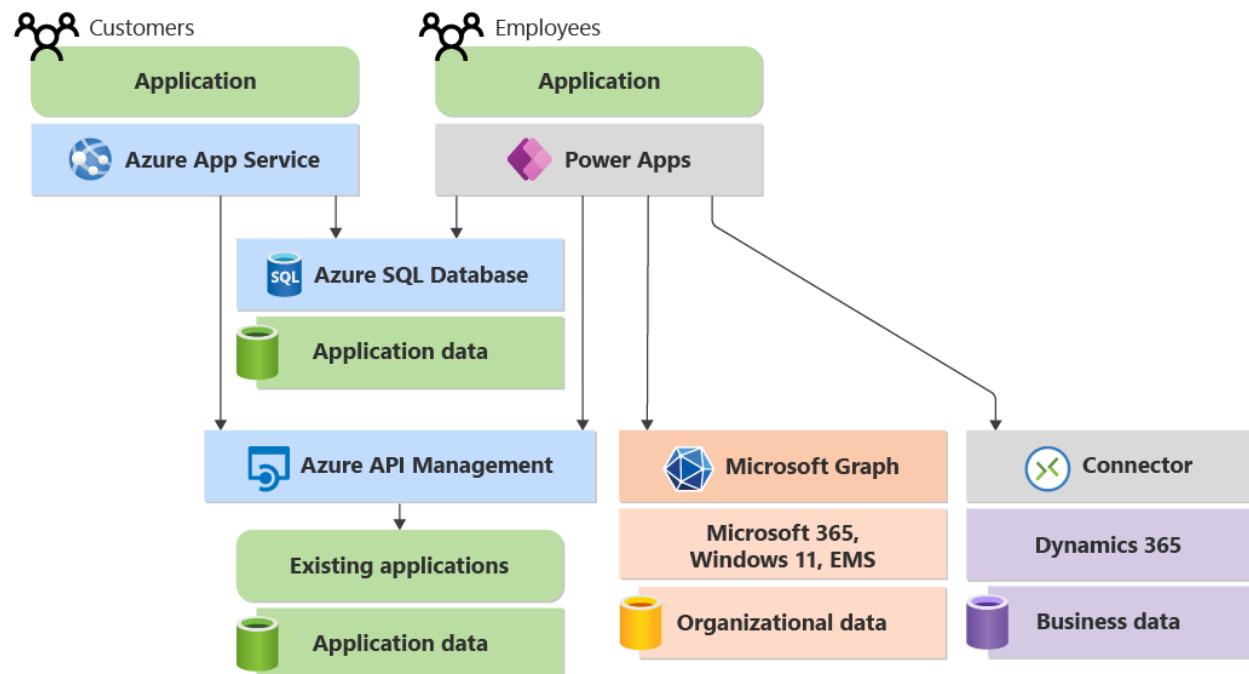


Figure 8: A low-code application can use a Power Platform connector to access Dynamics 365.

Power Platform provides various connectors for accessing Dynamics 365. One of the most widely used is the Microsoft Dataverse connector, which provides access to the data in Dynamics 365 Sales, Dynamics 365 Customer Service, and other Dynamics 365 offerings. This connector takes advantage of the fact that much of Dynamics 365 is built on Dataverse, making it simple to integrate with Power Platform. Although it's not shown in Figure 8, Dynamics 365 also exposes APIs for pro-code apps.

Connecting an enterprise application to Dynamics 365 business applications and data can be beneficial in many ways:

- A customer-facing application can integrate with Dynamics 365 Finance so that customers can work with invoices.
- A call center application, such as an employee-facing application created with Power Apps or Azure, can provide real-time predictions of how likely a customer is to switch to a competitor. Dynamics 365 Customer Insights includes pre-built machine learning models to provide this information.
- A recruiting application with both customer-facing and employee-facing components can integrate with Dynamics 365 Human Resources to store and access information about job openings and candidates.

There are connections between Dynamics 365 and other parts of the Microsoft Cloud. For example, a field technician can use Teams to notify users of Dynamics 365 Sales—sales professionals—about customers whose products are nearing end-of-life. The sales team can proactively tell the customers about replacement options.

Microsoft Industry Clouds

The Microsoft Cloud provides a broadly useful set of services for creating custom applications. Yet the solutions your organization creates probably aren't generic; they're specialized for your industry. What the Microsoft Cloud provides is helpful, but you'd also like more support for the industry-specific applications you need to build.

Microsoft Industry Clouds address this need. Each Industry Cloud builds on the Microsoft Cloud by adding industry-specific components. Here are some Industry Clouds:

- [Microsoft Cloud for Healthcare](#)
- [Microsoft Cloud for Manufacturing \(preview\)](#)
- [Microsoft Cloud for Retail](#)
- [Microsoft Cloud for Financial Services](#)
- [Microsoft Cloud for Nonprofit](#)

- Microsoft Cloud for Sustainability

Each Industry Cloud is designed to help you get the industry-specific value and compliance that you need. Each includes applications designed for a particular industry. For example, Cloud for Nonprofit includes a pre-built Power Apps solution for volunteer management, and Cloud for Healthcare includes an extensible chatbot so that customers can describe symptoms, then get information about related medical conditions and the type of doctor to consult.

Industry clouds also include components for application developers. Here are some examples:

- Cloud for Retail includes an intelligent recommendations service that suggests products to customers. This service is accessed via an API, so that developers can create customer-specific solutions that use it.
- Cloud for Healthcare includes a Fast Healthcare Interoperability Resources (FHIR) connector. This connector makes it easy for Power Platform solutions to connect to healthcare data that's accessible through the industry standard FHIR interface.
- Cloud for Financial Services, Cloud for Healthcare, and others provide pre-defined data types, called entities, for Dataverse and other datastores.

Building your own applications on the services that Microsoft Industry Clouds provide can help you create better and more compliant solutions in less time.

Next steps

See how successful enterprise application development leaders create and run secure applications by using Active Directory for identity and access management.

5. Create and run secure applications

Create and run secure applications

Article • 12/05/2023

ⓘ Note

This is article 5 of 6 in **Build applications on the Microsoft Cloud**.

Good security protects your systems, and therefore your organization, against accidental and intentional damage. It assures that only the right people can access your resources and minimizes the possibility that they can do inadvertent harm. It also warns you of dangers, violations, and other important security events.

This article discusses ways that Microsoft Cloud can help you secure your systems.

- [Use Microsoft Entra ID for a unified identity and access management solution](#)
- [Use Microsoft Sentinel to monitor and manage the security of your applications](#)

Use Microsoft Entra ID for a unified identity and access management solution

In our cloud-based world, employees and customers can access your custom applications from many devices in many different locations. Granting access to the right people, with the right restrictions, depends fundamentally on identity. Good security requires that each user prove their identity before they can access systems, and that they only access the resources they require to do their job.

Building the software to do this is hard. It requires specialists, and it takes time to get right, so you definitely don't want to build your own. Just as important, identity should be as simple to use as possible, both for your users and your developers. Ideally, you'd like a uniform way to manage identity throughout your environment.

This is what the Microsoft Cloud provides with [Microsoft Entra ID](#), the world's largest cloud identity service. If your organization uses any components of the Microsoft Cloud today, such as Azure, Power Platform, Microsoft 365, or Dynamics 365, you're already using Microsoft Entra ID. It's used throughout the Microsoft Cloud, giving your users a single identity for all of its components.

Your custom applications built on the Microsoft Cloud should also use Microsoft Entra ID. Figure 9 shows how this looks for our sample application.

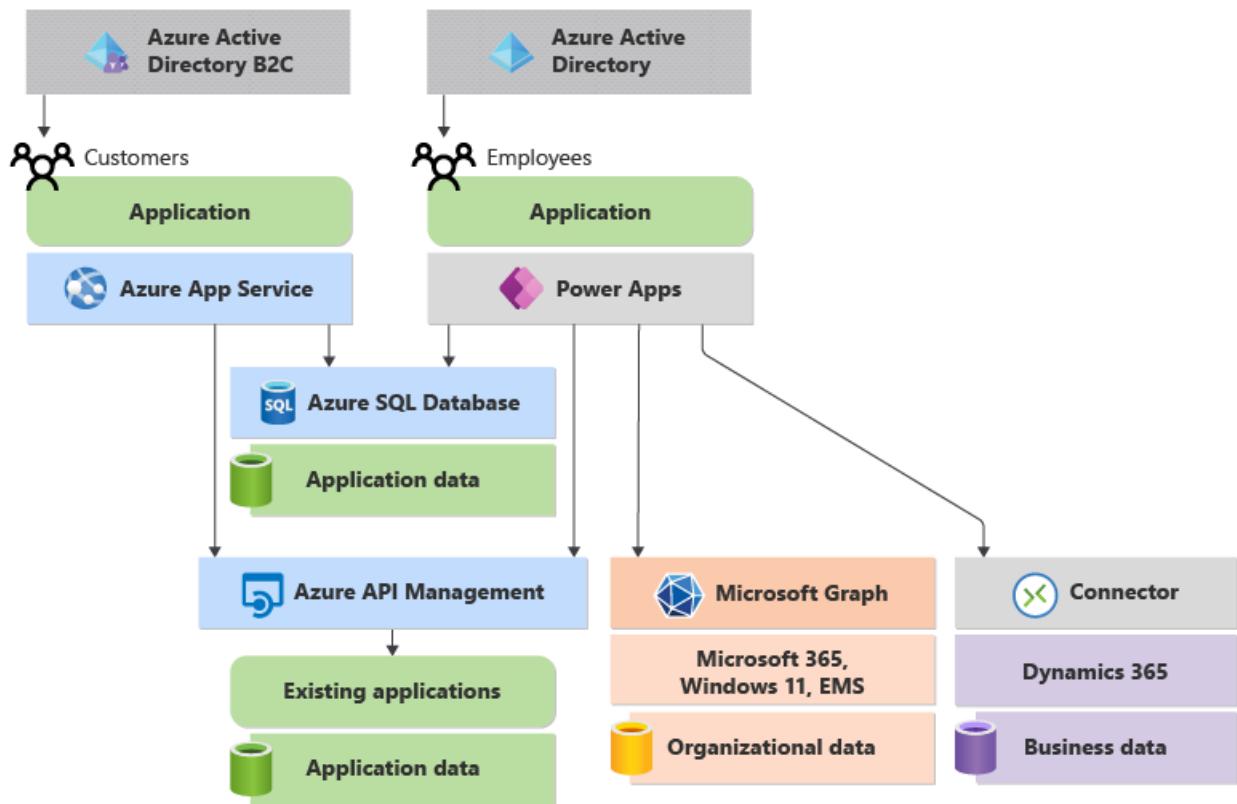


Figure 9: Microsoft Entra ID and Azure Active Directory B2C provide a common identity service for applications built on the Microsoft cloud.

As the figure shows, custom applications can use two related identity services:

- **Microsoft Entra ID**, which provides identities within the Microsoft Cloud. Employees who access your custom applications will typically use Entra ID to sign in and establish the identity they use to access all Microsoft Cloud services.
- **Microsoft Entra ID B2C**, which provides identities for external users. This service lets your customers create their own accounts or use existing public accounts from Microsoft, Google, Facebook, and others.

Using Microsoft Entra ID for identity brings several benefits:

- Having the same identity throughout the Microsoft Cloud makes life simpler for both developers and users of your applications. In the example shown in Figure 9, an employee can start by signing in to their organization's Microsoft Entra ID environment, known as a tenant. After they've done this, they can access the employee-facing component of the application that was created by using Power Apps. This application can call Azure API Management, Dynamics 365, and Microsoft Graph using the same identity, so the employee doesn't have to sign in again.
- Your developers can use the Microsoft identity platform in applications that they create. The libraries and management tools make it easier for developers to build applications that use identities from Microsoft Entra ID and elsewhere. To help do

this, the Microsoft identity platform implements industry standards such as OAuth 2.0 and OpenID Connect.

- Using Microsoft Entra ID and the Microsoft identity platform gives you control over how you use identity. For example, you can turn on support for multi-factor authentication in multiple applications built with the Microsoft identity platform by changing a single setting. Microsoft Entra ID also integrates with Microsoft's security tools for monitoring identity-based security threats and attacks.

Getting identity and access management right is a fundamental part of doing security well. Building applications on the Microsoft Cloud with Microsoft Entra ID makes this goal easier to achieve.

Use Microsoft Sentinel to monitor and manage the security of your applications

Everybody building applications today should assume that their software is targeted by attackers. Given this, your organization must continuously monitor and manage the security of your applications and the environment that they run in. The Microsoft Cloud provides several tools for doing this.

One of the most important of these is [Microsoft Sentinel](#). Sentinel provides security information and event management (SIEM), letting you capture and analyze a wide range of security-related data. It can also respond automatically to threats, providing security orchestration, automation, and response (SOAR). Sentinel can help your organization find and fix security problems more effectively.

Sentinel's broad reach encompasses the Microsoft Cloud and beyond through a large set of connectors. These connectors let Sentinel interact with many other services and technologies. Among the most important of these are the Microsoft Defender tools, including:

- [Microsoft Defender for Cloud](#), which helps your organization understand and improve the security of your Azure applications. It can also protect specific cloud services such as Azure Storage.
- [Microsoft 365 Defender](#), which provides components such as:
 - [Microsoft Defender for Office 365](#), which guards Exchange and other aspects of Office 365.
 - [Microsoft Defender for Identity](#), which monitors Active Directory to detect compromised identities and other threats.
 - [Microsoft Defender for Cloud Apps](#), which acts as a cloud access security broker between the users in your organization and the cloud resources that they use. It

helps you better understand which apps you use, both in the Microsoft Cloud and elsewhere, and who is using them.

Microsoft Sentinel can also import Office 365 audit logs, Azure activity logs, and other security relevant information within the Microsoft Cloud. Sentinel can also access security related information from many other sources provided by a diverse set of vendors. Once you've connected Sentinel to your information sources, you can analyze the data to understand security incidents and respond to them.

Security isn't a simple topic. Because of this, Microsoft provides Microsoft Sentinel and other security offerings to address this area. All these technologies work together to improve the security of applications running on the Microsoft Cloud.

Next steps

See a summary of **Build applications on the Microsoft Cloud** and find out how to learn more about succeeding as an enterprise application development leader.

[6. Summary](#)

Summary - Build applications on the Microsoft Cloud

08/12/2025

! Note

This is article 6 of 6 in **Build applications on the Microsoft Cloud**.

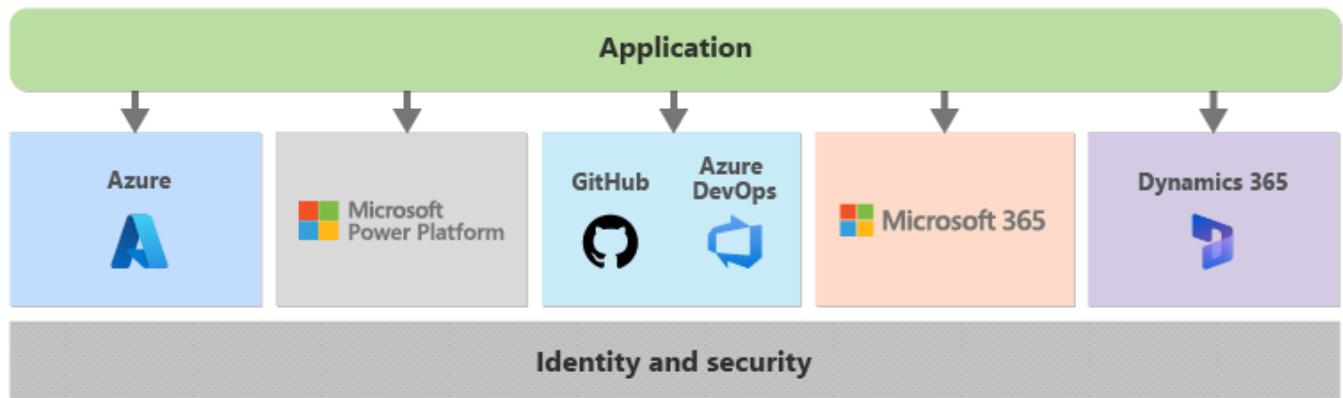


Figure 10: The Microsoft Cloud provides a unified set of services that applications can use.

The Microsoft Cloud, shown once more in Figure 10, provides a broad set of useful services to support custom applications. Building on this foundation can help you:

- Create better applications in less time with pro-code development on Azure and low-code development on Power Platform.
- Get the most value from technical talent with fusion development and an integrated low-code and pro-code development process that includes GitHub and Azure DevOps.
- Integrate new applications with existing solutions by using Azure API Management, Microsoft Graph, and Dynamics 365.
- Create and run secure applications by using Active Directory for identity and access management.

Use Azure Sentinel for security information and event management (SIEM), and for security orchestration, automation, and response (SOAR).

To learn more about building your applications on the Microsoft Cloud, visit our [Microsoft Cloud resources collection](#). It provides design guides, solutions, learning resources, and product documentation to help get you started.

Related resources

- These are the other articles in **Build applications on the Microsoft cloud:**
 - [1. Introduction](#)
 - [2. Create and deploy more applications in less time](#)
 - [3. Get the most value from technical talent](#)
 - [4. Integrate new applications with existing solutions](#)
 - [5. Create and run secure applications](#)
- [Video: Building Real-time Collaborative Apps with Azure, Microsoft 365, and Power Platform](#) ↗

Adopt, extend and build Copilot experiences across the Microsoft Cloud

Article • 07/12/2024

Copilot is an AI assistant powered by Large Language Models (LLMs) that offer innovative solutions across the Microsoft Cloud. They aim to enhance productivity, creativity, and data accessibility while providing enterprise-grade data security and privacy features. They bridge the gap between complex tasks and user-friendly solutions, providing a natural language conversational interface to interact with data, create automations, build applications, and even assist with coding tasks. Their integration within various Microsoft platforms and products provides a more interactive and efficient digital workspace.

The core value of Copilot stems from its ability to interpret natural language commands, enhancing the user experience by making it more intuitive and user-friendly. From answering queries in Microsoft Copilot, assisting with data navigation and analysis in Copilot in Microsoft Fabric and Copilot for Microsoft 365, aiding in app and automation creation in Copilot in Power Apps and Copilot in Power Automate, to suggesting code in GitHub Copilot, they allow users to have an AI-powered assistant at their fingertips. Copilot makes complex tasks more manageable, fostering a collaborative environment for both individual users and enterprises.



Copilot experiences across the Microsoft Cloud

Adopt Copilot	Extend Copilot	Build your own Copilot	
Copilot for Dynamics 365 Copilot for Microsoft 365 Copilot in Bing Copilot in Power Apps Copilot in Windows Github Copilot Many more...	Through low-code development Through plugins, message extensions, or connectors	With customizable Azure AI services through Azure AI Studio AND With pre-built Azure AI services through Microsoft Copilot Studio	Custom code Low code

Users can effortlessly adopt Copilot to streamline their workflows, while developers or creators have the flexibility to extend Copilot by integrating custom data. The core features that Copilot is built upon can also be used to build custom copilot solutions, which can be seamlessly integrated into new or existing applications.

The following table shows differences between the adopt, extend, and build options showcasing the breadth of possibilities that Copilot offers.

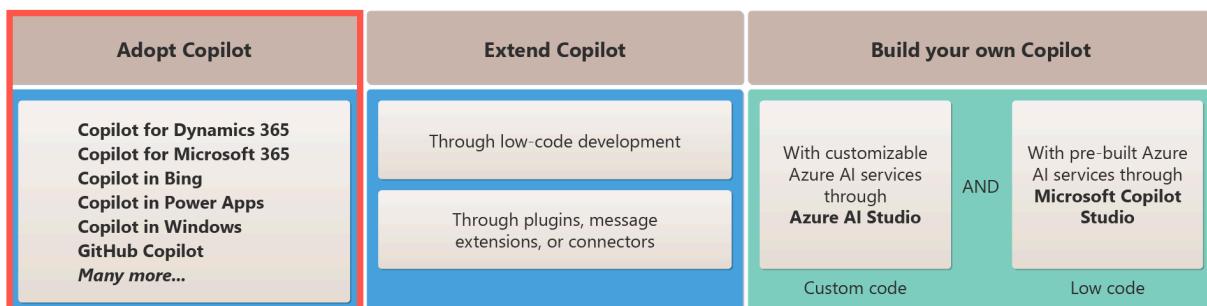
	Adopt Copilot	Extend Copilot	Build your own Copilot
Development effort	Low effort	Medium effort	Large to XL effort (depends on your data)
Data source	In Microsoft systems such as your Microsoft tenant	In Microsoft systems plus 3rd party data sources	Spans multiple data stores, services and apps inside and outside your Microsoft tenant
Users of Copilot	Users in your Microsoft tenant	Users in your Microsoft tenant	Users in your Microsoft tenant or external customers
Customization (temperature, max_tokens, metaprompts)	Low	Medium	High
Scenario	Copilot for Microsoft 365: "Show me all the upcoming trips on my calendar"	Copilot in Microsoft Teams: "Book a flight to Seattle" through the Flight API	Copilot for custom employee travel using Azure OpenAI Service or Microsoft Copilot Studio

Adopt Copilot

Microsoft offers a variety of Copilot assistants that can be adopted to enhance user productivity and creativity. Copilot is integrated into various Microsoft platforms and products, providing a more interactive and efficient digital workspace.



Copilot experiences across the Microsoft Cloud



Copilot assistants discussed in this section include:

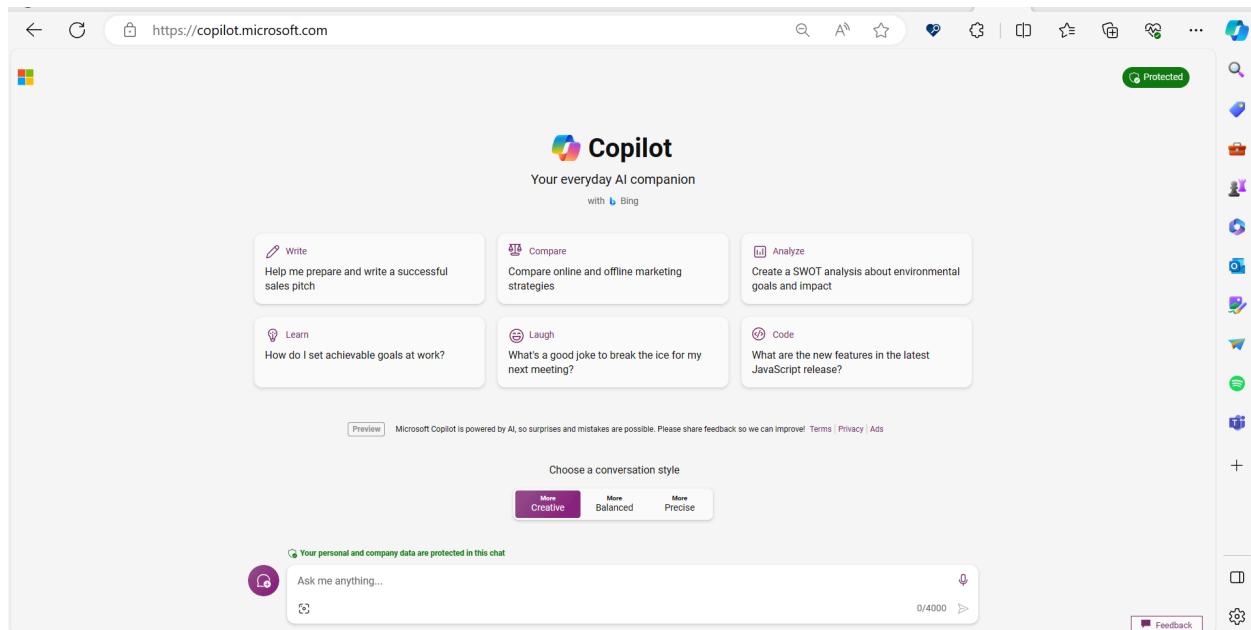
- [Microsoft Copilot](#)

- Copilot for Azure
- Copilot for Dynamics 365
- Copilot for Microsoft 365
- Copilot for Service
- Copilot in Microsoft Fabric
- Copilot in Power Apps
- Copilot in Power Automate
- Copilot in Windows
- GitHub Copilot
- Microsoft Security Copilot

Microsoft Copilot

Microsoft Copilot is an AI-powered chat assistant designed to aid users in web browsing and much more. It can answer both simple and complex queries, assist with research, and provide summaries of various content such as articles, books, or events. It can also offer product comparisons, find comprehensive answers, provide inspiration, generate images, and much more. Access Microsoft Copilot by visiting

<https://copilot.microsoft.com>. You can also access Microsoft Copilot by visiting <https://bing.com> and selecting "Chat" at the top of the page. Microsoft Edge users can access Microsoft Copilot by selecting the "Copilot" icon in the top right corner of the browser.



Enterprise users can also access Microsoft Copilot with commercial data protection which includes a secure AI-powered chat, ensuring user and business data safety. The system doesn't retain chat data and restricts data access, ensuring privacy. User data isn't used for model training ensuring data integrity. Copilot with commercial data protection offers accurate, verifiable, and visual answers from web data, aligning with

Microsoft's Responsible AI principles, making it a secure and efficient tool for modern enterprises.

Learn more:

- [Microsoft Copilot ↗](#)
- [Copilot with commercial data protection](#)
- [Privacy and protections](#)

Copilot for Azure

Copilot for Azure helps you design, operate, optimize, and troubleshoot your cloud infrastructure and services. Combining the power of cutting-edge large language models (LLMs) with the Azure Resource Model, Copilot for Azure enables rich understanding and management of everything that's happening in Azure, from the cloud to the edge.

Learn More:

- [Microsoft Copilot for Azure](#)

Copilot for Dynamics 365

Copilot for Dynamics 365 provides AI assistance to boost the productivity and efficiency of sales, support, supply chain management, finance, marketing and other teams involved with your business operations. It provides a chat interface for quick summaries of sales opportunities and leads, updates, meeting preparations, account-related news, and much more. Users can seamlessly integrate Copilot with Outlook and Teams for better data management and utilize its generative AI capabilities for record summarization and email content generation.

Learn More:

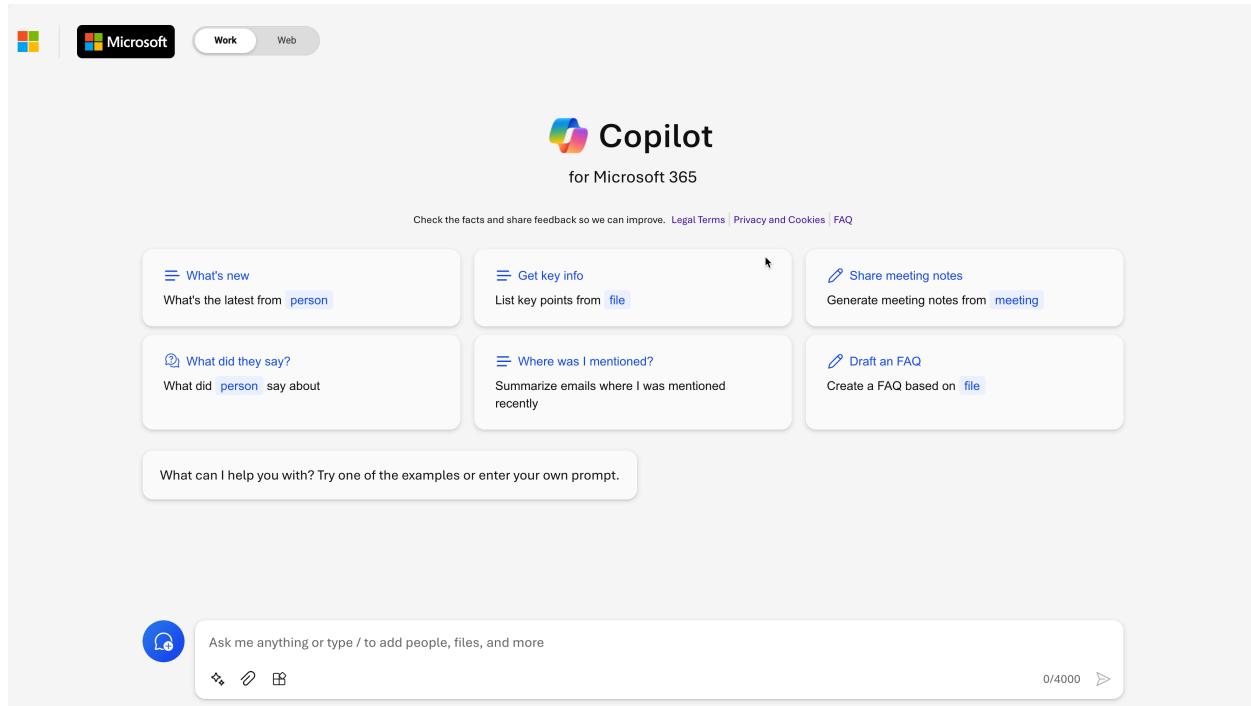
- [Copilot for Dynamics 365 Overview](#)

Copilot for Microsoft 365

Copilot for Microsoft 365 combines Large Language Models (LLMs) with Microsoft Graph, linking your organization's data with Microsoft 365 apps to create a new way of working. Users can easily find and navigate business data, analyze trends, and create insightful reports using simple natural language commands. This is a major upgrade for productivity, as it's integrated across the Microsoft 365 ecosystem, making tasks quicker

and easier to complete. Plus, with built-in enterprise-grade security, privacy, and Responsible AI, it ensures a safe, secure, and compliant digital workspace.

In addition, Copilot functionality is also available directly in Microsoft 365 apps. The integrated Copilot functionality provides users with a conversational interface to jumpstart their work and offer suggestions and insights to enhance productivity and creativity.



Learn more:

- [Copilot for Microsoft 365 Documentation](#)
- [Use Restricted SharePoint Search with Copilot for Microsoft 365 ↗](#)

Copilot for Service

Copilot for Service synthesizes vast amounts of data already available from an organization's trusted knowledge sources to provide relevant, timely guidance to agents in their flow of work. Copilot for Service provides flexibility for organizations to quickly begin realizing the benefits of generative AI by extending their existing investments in CRM and contact center solutions—including Dynamics 365, Salesforce, ServiceNow and more.

Learn More:

- [Copilot for Service Documentation](#)

Copilot in Microsoft Fabric

Copilot in Microsoft Fabric brings a new way to transform and analyze data, generate insights, and create visualizations and reports.

Copilot for Power BI in Microsoft Fabric is infusing the power of large language models into Power BI at every layer to help users get more done and create additional value from their data. Users can create and tailor reports in seconds, generate and edit DAX calculations, create narrative summaries, and ask questions about their data, all in conversational language. It enables you to quickly create reports with just a few clicks. It saves you hours of effort building your report pages. Copilot provides a summary of your dataset and an outline of suggested pages for your report.

Copilot for Power BI does the following:

- A summary of the dataset.
- A report outlines of suggested pages for your report: for example, what each page in the report will be about, and how many pages it will create.
- The visuals for the individual pages.
- A summary of the current page.

Learn more:

- [Overview of Copilot in Microsoft Fabric](#)
- [Overview of Copilot for Power BI](#)
- [Copilot for Power BI Demo ↗](#)
- [Analyzing Snapshot Serengeti data with Microsoft Fabric ↗](#)

Copilot in Power Apps

Copilot in Power Apps enables app creators to build applications more efficiently than ever. It enables the construction of an app, along with its data, by articulating the requirements through a conversational, natural language approach over several steps. This feature ensures a Copilot-driven experience from the initial screen, offering users a more conversational and less click-intensive interaction as they build apps.

Learn more:

- [Copilot in Power Apps Overview](#)
- [Build a canvas app for a real estate solution with Copilot in Power Apps](#)

Copilot in Power Automate

Copilot in Power Automate simplifies automation creation using natural language expressions. Users describe what they need through conversation, and Copilot assists by

understanding intent, setting up connections, applying necessary parameters, and making requested changes to the flow. It can also be used to answer queries about the flow that a user is creating.

Learn more:

- [Get started with Copilot in Cloud Flows](#)
- [Build flows for a real estate solution using Copilot in Power Automate](#)

Copilot in Windows

Copilot in Windows is an AI-powered intelligent assistant designed to enhance user efficiency and creativity. It aids in retrieving answers and inspirations from the web, supporting creative tasks, and aiding focus on the current task. Users can adjust PC settings, organize windows, and initiate creative projects with Copilot's assistance. It's easily accessible, ready with a keystroke, and extends support both online and within Windows apps, making it a user-friendly tool for various tasks.

Learn more:

- [AI with Copilot in Windows ↗](#)

GitHub Copilot

GitHub Copilot is an AI developer tool that assists in the coding process by suggesting code as the user types. It serves as a programming aid, helping to streamline coding tasks and explore coding solutions efficiently. Integrated within the GitHub platform, it provides a supportive environment for developers to tackle programming challenges and enhance their coding workflow. Additionally, GitHub Copilot can be used to learn new programming languages or frameworks by providing real-time code suggestions based on developer input.

Learn more:

- [GitHub Copilot Overview ↗](#)

Microsoft Security Copilot

Microsoft Security Copilot is an AI-powered security analysis tool that empowers defenders to move at the speed and scale of AI, combining advanced large language models with a security-specific model from Microsoft. It integrates insights and data from security tools and delivers guidance that's tailored to your organization. Security Copilot accepts natural language inputs and includes a pinboard section for co-workers

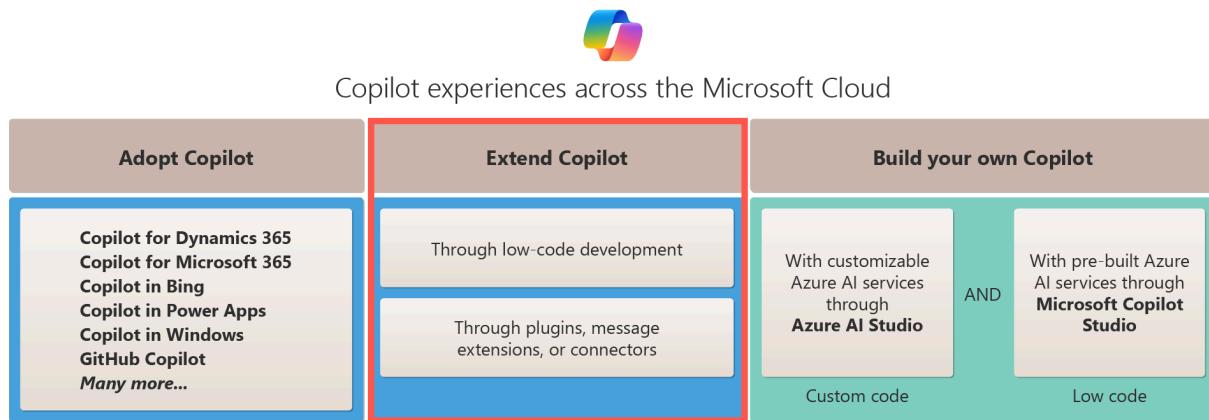
to collaborate and share information. It also surfaces prioritized threats in real-time while anticipating a threat actor's next move with continuous reasoning based on Microsoft's global threat intelligence.

Learn more:

- [Microsoft Security Copilot Overview](#)
- [Get started with Microsoft Copilot for Security](#)

Extend Copilot

Copilot offers a comprehensive range of features right out of the box, yet many users depend on external tools, services, and data to carry out tasks. To bridge this gap, Copilot can be extended using plugins, connectors, or message extensions.



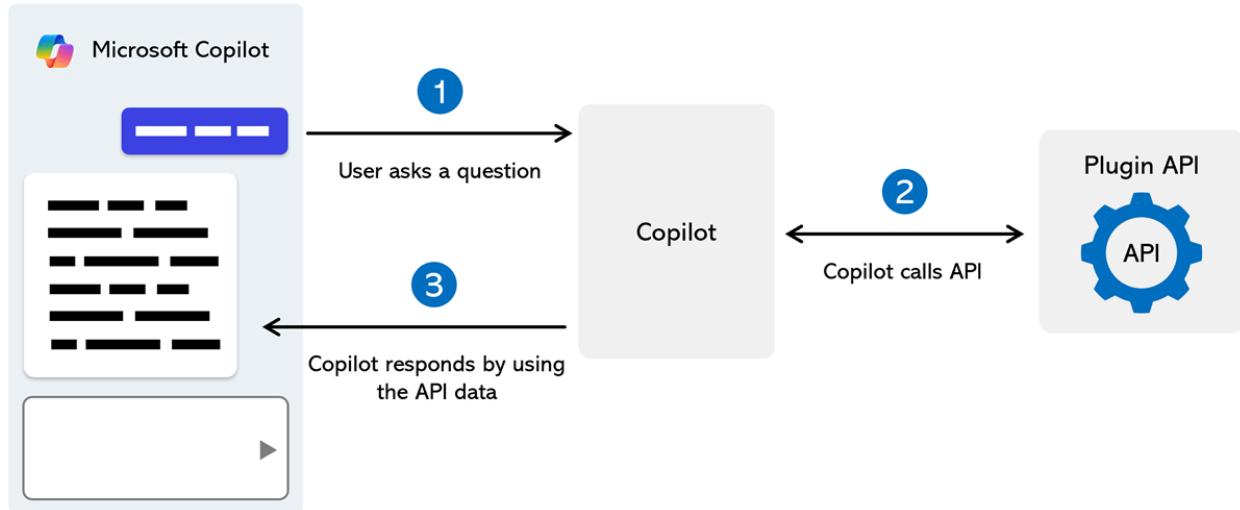
Benefits of extending Copilot

By extending Copilot, developers can integrate external data directly to minimize user context shifts while enhancing productivity and collaboration. Copilot can also be extended to provide a new way for developers to surface external data directly in Microsoft products that users use every day. For instance, enterprises and ISVs can build plugins to integrate their own line-of-business APIs and data directly into Copilot.

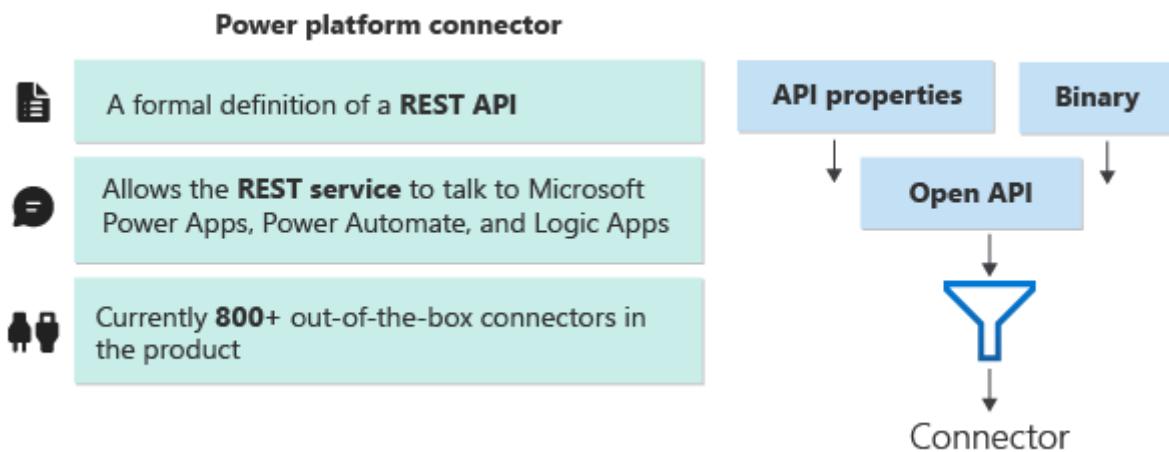
Adding plugins, connectors, or message extensions to Copilot helps users make the most out of these AI assistants. Users can more easily discover and work with enterprise data, tools, and services using natural language, which improves productivity and enhances teamwork.

Extending with Plugins

Plugins allow users to interact directly with APIs and services enhancing the capabilities of Copilot and broadening its range of functionality by pulling in external data. For example, plugins can enable Copilot to retrieve real-time data, such as the latest news coverage on a product launch, or knowledge-based information such as a team's project items.



One example of plugins is Power Platform connectors which offer a set of prebuilt actions and triggers for developers to build their own apps and workflows. They can be used to extend Copilot for Microsoft 365 and Copilot in Power Platform experiences with data from external apps.



Developers can also create plugins that use schemas to add custom functionality to Copilot experiences by connecting their own application data to Microsoft Copilot. Plugins enable a Copilot experience to interact with your own APIs, enhancing the experience with the ability to perform a wide range of actions.

Learn more:

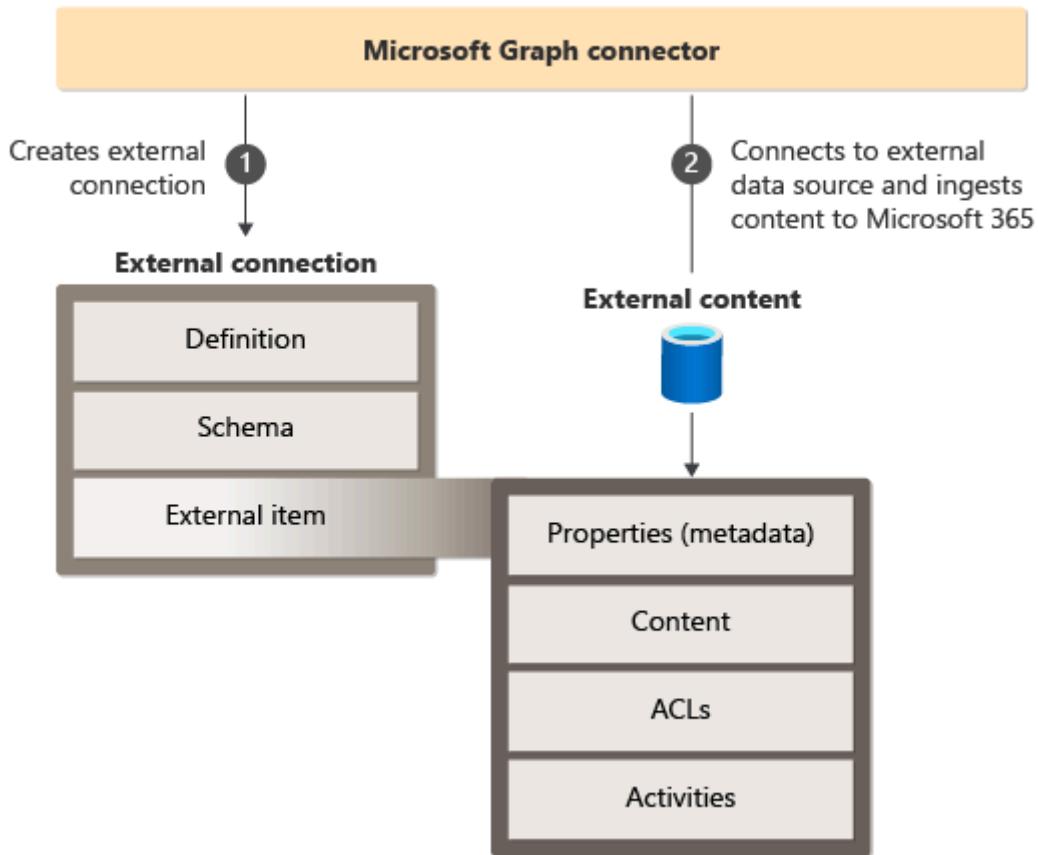
- [Power Platform Connectors Overview](#)
- [Create a Custom Power Platform Connector from Scratch](#)

- Overview of Plugins for Microsoft Copilot
- Copilot for Microsoft 365 Copilot Plugins Samples ↗

Extending with Microsoft Graph connectors

Microsoft Graph connectors import external content into Microsoft 365. Once the content is imported, all of Microsoft 365 including search, context IQ, microsoft365.com, and Copilot for Microsoft 365 can access the content. Microsoft Graph connectors are responsible for three key tasks:

1. Creating an external connection in Microsoft 365
2. Defining a schema for the external content
3. Managing external content that's imported into Microsoft 365



Learn more:

- Build your First Custom Microsoft Graph Connector
- Publish your Custom Microsoft Graph Connector
- Microsoft Graph Connectors Samples ↗

Extending with message extensions

Message extensions allow users to interact with enterprise services through Microsoft Teams chats. They can search or initiate actions in an external service and post that information, in the form of adaptive cards, into a Teams message. By using message extensions, you can extend the built-in functionality of Copilot for Microsoft 365.

Learn more:

- [Message Extensions Overview](#)
- [Build Message Extensions using Bot Framework](#)
- [Northwind Inventory Plugin for Copilot for Microsoft 365](#) ↗

Extending with Microsoft Copilot Studio

Microsoft Copilot Studio is a low-code tool that helps create solutions for understanding and gaining insights from your business data and activities. Through easy drag-and-drop actions, developers and creators can add essential logic and data connections, allowing your custom data to be integrated with Copilot for Microsoft 365.

Learn more:

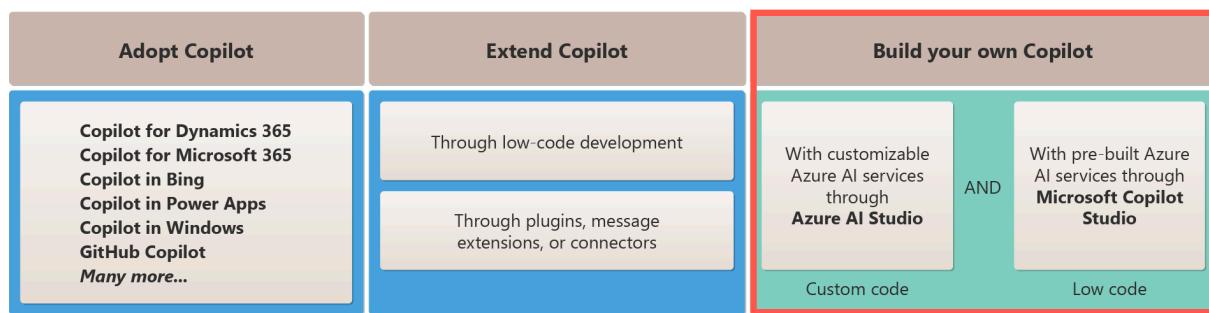
- [Microsoft Copilot Studio Overview](#)

Build your own Copilot

In addition to adopting and extending Microsoft Copilot, a custom AI copilot can be created to create a personalized conversational AI experience for users using Azure OpenAI Service, Azure AI Search, Microsoft Copilot Studio, and other Microsoft Cloud technologies. A custom copilot can integrate company data and documents, retrieve real-time data from external APIs, and be embedded in company applications. For example, a custom copilot can be built to provide a chat interface to access help desk tickets and knowledge base articles. Rather than using traditional search techniques or navigating across multiple applications, users can use the conversational UI provided by the custom Copilot to find the information they need.



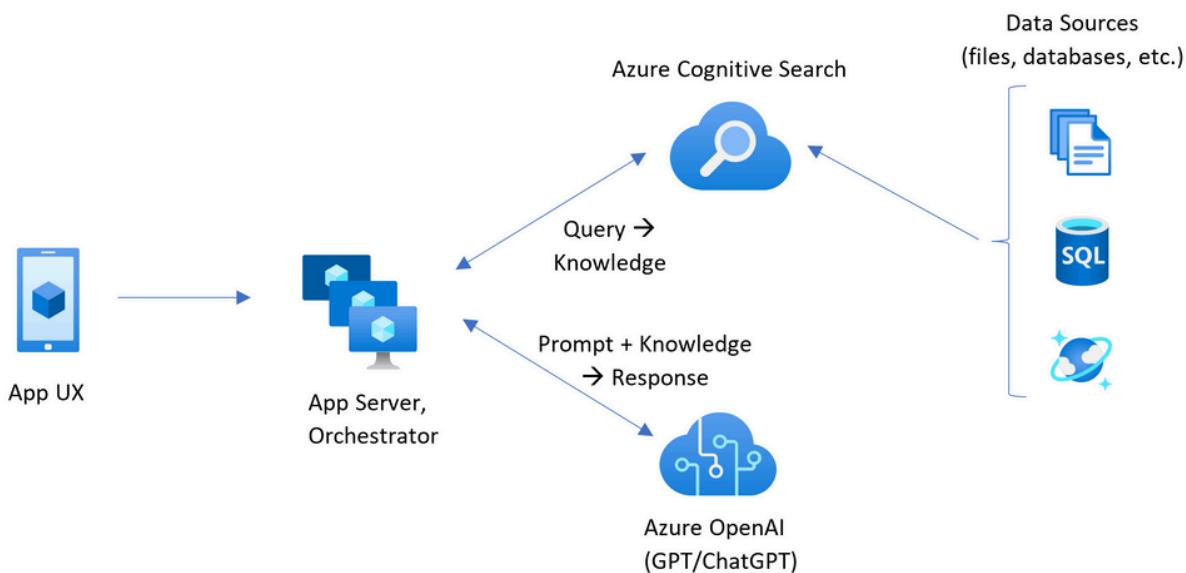
Copilot experiences across the Microsoft Cloud



Azure AI Studio

Azure AI Studio brings together capabilities from across multiple Azure AI services. With Azure AI Studio, you can evaluate large language model (LLM) responses and orchestrate prompt application components with prompt flow for better performance.

Companies can integrate their own data with Azure AI Search and Azure OpenAI models to create an advanced conversational AI platform that can be embedded in their applications. This enables users to interact with company data and documents using natural language, making it easier to find information and complete tasks.



To chat with your own data using Azure OpenAI and Azure AI Search in Azure AI Studio, see [Deploy a web app for chat on your data](#) and [Build your own copilot using prompt flow](#).

Learn more:

- [Azure AI services Documentation, Samples, and Resources](#)
- [Azure AI Studio Documentation](#)

- [Quickstart: Chat with Azure OpenAI Models Using Your Own Data](#)
- [Integrate OpenAI, Communication, and Organizational Data Features into a Line of Business App](#)

Solution accelerators and samples

Accelerators and samples are available to help you get started building a custom copilot that integrates with your own data and applications.

Learn more:

- "Chat with your data" Solution Accelerator ↗
- [Azure Chat Solution Accelerator powered by Azure OpenAI Service ↗](#)
- [ChatGPT + Enterprise data with Azure OpenAI and Azure AI Search ↗](#)
- [ChatGPT-Powered and Voice-Enabled Assistant ↗](#)
- [GPT Powered Customer Service with Chat/Voice/Video ↗](#)

Microsoft Copilot Studio

In addition to extending Copilot for Microsoft 365, Microsoft Copilot Studio enables you to build custom Copilot solutions using your own data. Microsoft Copilot Studio is a low-code tool that enables the creation of robust AI-driven chatbots and GPTs capable of handling an array of interactions, from addressing standard inquiries to navigating intricate dialogues for issue resolution. Copilot Studio not only simplifies the process of building intelligent chatbots, but also seamlessly integrates across a variety of digital platforms such as websites, mobile apps, Facebook, and Microsoft Teams.

Learn more:

- [Microsoft Copilot Studio Overview](#)
- [Quickstart: Create and deploy a Microsoft Copilot Studio Copilot](#)

Copilot for Dynamics 365

Article • 07/12/2024

Copilot for Dynamics 365 provides AI assistance to boost the productivity and efficiency of sales, support, supply chain management, finance, marketing and other teams involved with your business operations. It provides a chat interface for quick summaries of sales opportunities and leads, updates, meeting preparations, account-related news, and much more. Users can seamlessly integrate Copilot with Outlook and Teams for better data management and utilize its generative AI capabilities for record summarization and email content generation.

Copilot in Dynamics 365 Business Central

With Copilot in Dynamics 365 Business Central, small and medium-sized organizations empower their workers with intuitive AI tools that boost creativity and save time. Copilot provides the following benefits:

- **Bringing new products to market faster:** Start by uploading a product picture that is automatically matched to a product category and attributes, reducing data entry. Banish writer's block by having Copilot draft compelling product descriptions based on attributes such as color, material and size. Tailor descriptions to your brand by adjusting tone of voice, format and which quality to emphasize. With just a few clicks, your products are published to your online store and are ready to transact, using Business Central's integration to Shopify, a leading eCommerce platform.
- **Reduce tedious accounting:** Copilot assists with the recurring task of reconciling bank account statements that typically takes hours. After uploading a bank statement file or pulling the latest statement digitally from your bank, Copilot puts AI to work to automatically match transactions known to Business Central, and suggests an appropriate ledger account in which to register any remaining transactions.
- **Extend Copilot with custom capabilities:** Developer tools for Copilot in Business Central empower our community developers to build powerful, generative AI experiences for any role or industry, and deploy them to their favorite customers or distribute them as Add-on apps at scale on Microsoft AppSource. The toolkit includes signature Copilot user experiences, guard-rails that safeguard your customers and their data, and APIs that simplify coding against Azure OpenAI Service.

Learn more:

- [Copilot in Business Central ↗](#)
- [Sign up for a free trial ↗](#)
- [How to drive sales with AI-generated product descriptions](#)
- [How to AI-powered item marketing text with Copilot](#)
- [How to create AI-generated marketing text with Copilot](#)

Copilot in Dynamics 365 Customer Service & Microsoft Copilot for Service

Copilot in Dynamics 365 Customer Service empowers agents to deliver exceptional customer care. Dynamics 365 Copilot drafts contextual answers to queries in both chat and email, in addition to providing an interactive chat experience over knowledge bases and case history so this AI-powered expertise is always available to answer questions. We're also making it easier for customer service departments to build virtual agents in minutes with conversation boosters in Power Virtual Agents, which harnesses the power of Azure OpenAI Service and Bing to provide answers from company websites and internal knowledge bases that you choose.

Sellers spend much time managing complex customer relationships, and many sales teams need help to scale seller experiences. Seller productivity in terms of time management and efficiency is critical to success. Copilot in Dynamics 365 Sales experiences focus on enhancing seller productivity and effectiveness in their flow of work by enabling them to automate the sales process and augment seller actions and decisions with AI-powered insights and actions. Copilot features help sellers save time, boost productivity, lighten workloads, and stay focused on connecting with customers and closing deals.

Copilot in Dynamics 365 Customer Service offers the following capabilities:

- Summarize a case and conversation
- Respond to questions
- Compose an email
- Draft a chat response
- Ask a question about knowledge base sources and trusted domains

Microsoft Copilot for Service offers the following capabilities:

- Connect to existing contact center and CRM solutions, with out-of-the-box integrations for Dynamics 365, Salesforce, ServiceNow, and Zendesk
- Quickly bring agents up to speed on email threads within Outlook by providing summaries, as well as accelerate responses to customers by providing draft emails

- In Teams, agents can stay on top of meetings with customers or subject matter experts by viewing AI-powered meeting recaps with summaries of key discussion points, sentiment, and next steps
- Agents can use Copilot to get the latest account and case information from CRM systems

Learn more:

- [Copilot in Dynamics 365 Customer Service](#)
- [Microsoft Copilot for Service](#)↗

Copilot in Dynamics 365 Customer Insights

With Copilot in Dynamics 365 Customer Insights, marketers can curate highly personalized and targeted customer segments by having a dialogue with their customer data platform using natural language. Marketers can receive suggestions about additional segments that may not have been previously considered. This new capability can handle complex calculations and match customers that fit a select profile.

Copilot in Dynamics 365 Customer Insights - Journeys

Using Copilot in Dynamics 365 Marketing, marketers can describe their customer segment in their own words to create a target segment with the query assist feature. Marketers can also use Dynamics 365 Copilot to effortlessly get inspiration for fresh email campaign content based on a simple request. Copilot makes suggestions based on key topics entered by the marketer, the organization's existing marketing emails, as well as from a range of internet sources to increase the relevance of generated ideas.

Learn more:

- [Use copilot to style your emails with AI-assisted themes](#)
- [Use copilot to kickstart email creation](#)
- [Use copilot to refresh and perfect your message](#)
- [Use copilot to create journeys](#)
- [Use copilot to target the right customers using query assist](#)

Copilot in Dynamics 365 Field Service

The AI-generated work order recap in Dynamics 365 Field Service provides a summary of a work order and its related data. Designed to help service managers, dispatchers,

and frontline workers quickly understand the state of a work order and the next steps to take, the recap summarizes the work order's history, future actions, arrival times, work criticality, required parts, and other contextual information. The Outlook add-in also provides copilot capabilities to assist with work order creation and scheduling in Outlook.

Learn more:

- [Work order recap with Copilot in Field Service](#)
- [Work order creation with Copilot](#)

Copilot in Dynamics 365 Project Operations

Dynamics 365 Project Operations Copilot empowers project managers to swiftly create project plans for new engagements in a matter of seconds. By describing project details in natural language, they can save valuable hours that would otherwise be spent on meticulous planning. These plans can be easily customized to align with the specific project requirements.

Copilot simplifies the creation of project status reports, reducing the time spent on research and writing. These reports integrate concise summaries of scheduling and financial data, providing valuable insights into project progress, financial performance, and schedule adherence. Project managers can generate comprehensive status reports quickly and efficiently, allowing for a more strategic allocation of their time.

Copilot excels in continuous risk identification and mitigation. It assists project managers in identifying and addressing potential risks that could impact project timelines and budgets. For instance, you can instruct Copilot to scan across your active projects to pinpoint common risk factors, such as significant delays or budget overruns.

With Copilot, project managers can improve efficiency, reduce risks, and focus on strategic and value-added activities.

Learn more:

- [Project Management Copilot Overview](#)

Copilot in Dynamics 365 Sales & Microsoft Sales Copilot

Copilot in Microsoft Dynamics 365 Sales and Microsoft Sales Copilot helps sellers dramatically reduce the time they spend on clerical tasks. AI helps write email responses

to customers and can even create an email summary of a Teams meeting in Outlook. The meeting summary pulls in details from the seller's CRM such as product and pricing information, as well as insights from the recorded Teams call. With sellers spending as much as 66% of their day checking and responding to emails, this presents a significant business upside to give the seller more time with their customers.

Copilot in Dynamics 365 Sales offers the following capabilities:

- Record summarization
- Record catch up
- Meeting preparation
- Email assistance
- News updates

Microsoft Sales Copilot offers the following capabilities:

- AI-generated lead and opportunity summaries to streamline your sales process and close deals faster
- A customer meeting preparation summary view that includes auto-generated summaries of previous meetings, recent notes, highlights of issues and concerns, and CRM information
- Real-time tips and suggested answers during Teams meetings, prompted by the customer's mention of a competitor or brand
- Customer email summary and follow-up, with contextual replies and actions based on Outlook emails and CRM data

Learn more:

- [Copilot in Dynamics 365 Sales](#)
- [How to enable configure Copilot in Dynamics 365 Sales](#)
- [Microsoft Sales Copilot](#)

Copilot in Dynamics 365 F&O Help & Guidance

Dynamics 365 Finance and Operations provides a rich set of capabilities for supporting organizations' business processes. The breadth of functionality means that, as with most business applications, users require training to get started and need to acquire a fair amount of experience before they can become proficient. In the meantime, users must rely on the availability of experts, and support from partners or Microsoft along the learning journey. Now, with innovative in-product, AI-based, contextual help, users can quickly find the information they need to rapidly unlock the full potential of Dynamics 365 Finance and Operations.

Learn more:

- [Empower users with conversational product help and guidance](#)

Copilot in Dynamics 365 Commerce

Creating marketing and storytelling content for a product for e-commerce sites is key to driving customer engagement and sales, but can be time-consuming and challenging. The new copilot capability in Commerce is built for the product enrichment workflow in site builder. With Copilot for site builder, you can generate persuasive, compelling, and engaging product enrichment and marketing content for your e-commerce websites quickly and efficiently.

Copilot in Dynamics 365 Commerce offers the following capabilities:

- Jump-start the creative process: Copilot creates content that is based on basic product information such as name, description, attributes, and dimensions. You can use the generated content to spark further ideas.
- Optimize for search engines (SEO): Optimize your product marketing content for search engines so that it ranks higher in Internet search results and thereby helps generate more organic growth.
- Craft the tone of message: Select the voice that best fits your brand and audience. You can specify a unique tone for each product. The tone of message for a product can be adventurous, casual, luxurious, formal, informational, educational, and more.
- Craft a tailored message: Select the intended audience to tailor the message for the specific buyers, such as new parents, graduates, senior citizens, health-care workers, and more.
- Amplify with key highlights: Augment your content with key highlights specific to the product in a natural language-based and interactive manner. Generate engaging text snippets that help amplify key highlights for your product.
- Make basic modifications: Easily make basic text modifications using an inline rich text editor or by providing key prompts to format the text using natural language.

Learn more:

- [Generate Product Enrichment Content for E-commerce Sites with Copilot in Site Builder](#)

Copilot in Dynamics 365 Finance

Copilot in Dynamics 365 Finance enhances productivity for collections managers and improves collaboration with their customers.

The AI-generated summary of the customers' account, overdue invoices and payment history helps prioritize which customers need attention. Personalized customer communication using the AI-generated email saves time and increases overall efficiency by not having to navigate to multiple screens. This helps increase successful collection rates and proactively keeps customers in good standing.

Copilot in Microsoft Dynamics 365 Finance displays this AI-generated summary on the new Collections coordinator workspace. Powered by Microsoft Azure OpenAI's large language model, this will reduce the time reviewing collections details for customers. Collections coordinators can help their organizations to better cash flows by increasing more timely payments.

Learn more:

- [Collections Coordinator Summary](#)

Copilot in Dynamics 365 Supply Chain Management

Copilot in Microsoft Supply Chain Center, which Microsoft Dynamics 365 Supply Chain Management customers can access, will proactively flag external issues such as weather, financials and geography that may impact key supply chain processes. Predictive insights then surface impacted orders across materials, inventory, carrier, distribution network and more. Supply chain planners can then automatically draft an email generated by Dynamics 365 Copilot to alert impacted partners and mitigate potential disruptions before they happen.

In-App Help Guidance to provide users with immediate, contextual assistance directly within the application and user processes:

- **Draft precise responses to user queries:** Copilot utilizes the extensive repository of public documentation on Dynamics 365 Supply Chain Management features and grounds generative AI on the product documentation to craft precise responses to user queries
- **Context aware guidance:** Users can pose questions related to their current context and Copilot provides tailored and specific guidance.

On-hand inventory insights to help user make informed inventory decision such as for order promising:

- **Query on-hand inventory:** Users can use natural language to ask about inventory stock levels and availability just as easily as when chatting with a colleague. For example, you can simply type "Find all the 18-inch tires in my store," without needing to remember any exact specifications, IDs, or navigation paths.
- **Inquire Inventory:** Use a guided conversational experience to quickly identify the best available products for order inquiries without leaving the Microsoft 365 work context. Sales managers focus on communicating with customers and increasing sales. Having product inventory levels handy from Microsoft 365 helps sales managers please customers and close deals faster.
- **Enable third party apps or websites to query on-hand Inventory:** Natural language search APIs for Inventory Visibility can integrate with third-party e-commerce websites or apps. Customers can use them to build a chat feature for third-party apps or websites that helps customers find products, view availability, and view alternatives.

Copilot helps procurement specialists to make better purchasing decisions in procure-to-pay processes:

- **Manage purchasing decisions with Copilot:** The Copilot support for procurement is an advanced tool that empowers users to efficiently manage and assess changes to purchase orders on a large scale. By classifying changes by impact and risk associated with these changes, it greatly enhances procurement decision-making. With Copilot, users can swiftly identify high-impact or low-impact alterations, analyze downstream impact enabling prompt actions to mitigate potential risks.

Learn more:

- [Review and Accept Changes to Confirmed Purchase Orders](#)

Developing a commercial strategy for generative AI applications

Article • 09/26/2024

This article enables Independent Software Vendors (ISVs) and Software Development Companies (SDCs) to:

- Learn about different approaches to commercial strategy for generative AI applications
- Understand how other ISVs successfully designed and implemented a commercial strategy for their applications

Introduction

As ISVs consider generative AI use cases and how to develop them, it's critical to think about the commercial strategy of your solution from the very beginning. Aligning your development and commercial approaches from the outset can help you establish a more cohesive, and successful overall product strategy.

This article is intended to help you determine your application's commercialization strategy by providing a high-level view of the considerations that can impact your commercial model and real-world examples of commercial applications. This commercialization page can be useful as you evaluate use cases with the [business envisioning](#) page and select a technical approach in the [capability envisioning](#) page. While there's no one-size-fits-all approach to commercialization, understanding the strategies that successfully worked for others can help guide you in successfully monetizing your investments.

Let's walk through some initial challenges and considerations that your commercial strategy must address.

Commercialization challenges and considerations

When you align on the commercial strategy and monetization approach for your solution, there are several challenges to navigate:

- **Volatile demand and value perceptions** that can shift quickly as new products are brought to market
- **Highly variable and dynamic costs**, including development, deployment, and hosting costs
- **Market differentiation** that aligns firm profitability with customer affordability and keeps a path open for future scalability

As with any solution an ISV builds, it's important to establish a clear business plan and understand the potential revenue opportunity. These initial steps help ensure clear value alignment and drive a successful approach to commercialization.

To plan a successful commercial strategy, it can be helpful to examine your solution and context through multiple lenses. As you think about your solution's commercial strategy, there are three primary things to consider: value determination, costs, and your go-to-market approach. We discuss each of these in-depth later using a series of ISV example scenarios. The following considerations can help you think through the factors that go into choosing your commercialization strategy.

Focus area	Consideration	Ask yourself
Value determination	Business value	<ul style="list-style-type: none"> • What tangible and intangible benefits does this solution provide? • How can the value of this solution be clearly communicated and measured?
	End-user(s)	<ul style="list-style-type: none"> • Who are the primary users of this solution, and what are their specific pain points? • How can end users integrate this solution into their existing workflows?
Costs	Technical approach	<ul style="list-style-type: none"> • How are you building this solution? • How does the solution's architecture enable scalability to meet future growth?
	Costs	<ul style="list-style-type: none"> • What are the initial development and ongoing operational costs associated with this solution? • How do you plan to manage cloud costs and fluctuating usage?
Go-to-market	Go-to-market strategy	<ul style="list-style-type: none"> • How can you position your solution in the market to differentiate it from competitors? • What sales enablement tools and resources will you need to effectively promote and sell your solution?

Thinking through these considerations can help you address the challenges that many commercial efforts face. Let's briefly discuss how each of these focus areas can impact your commercial strategy.

Value determination

As you develop your commercialization strategy, it's critical to identify the value your application creates for you and your customer. Consider your objective as you identify your use case and build your application. Are you building brand awareness in a new

target market or exploring a supplemental revenue opportunity in an existing audience? Having a clear goal and success metric for your solution can help align your commercial strategy with your application. Through our conversations with ISVs, we find that many ISVs seek to achieve the following value from their offerings:

- **Revenue** - the solution is offered at a charge to customers and intended to generate ISV revenue streams
- **Market share** - the solution is intended to build brand awareness and increase ISV market share in a specific area

While solutions that focus on generating revenue provide obvious benefit for ISVs, there's still value in developing applications with the principal intention of expanding market share and brand awareness. For example, an ISV entering a new market may want to publish an initial series of solutions or features for free to generate interest and trust in their brand before a larger, revenue-focused release. However, regardless of your objective, aligning your solution with customer demands is critical for a successful commercial strategy.

Consider customer demand

Whether you're focusing on increasing market share or driving revenue, ISV solutions perform better with customers when they provide a clear value proposition to the customer. To effectively decide on a monetization and pricing strategy, it's essential to evaluate how much value your solution holds to your customer.

Consider how your solution helps the customer-what value does it provide for them? Typically, solutions provide value to customers in at least one of three ways:

- **Increase effectiveness** - enable customers with new capabilities that were previously not available
- **Improve efficiency** - reduce the time, labor, or resources required to complete tasks or operations
- **Generate revenue** - generate revenue directly by allowing your customer to market and/or monetize their own products

If your end customers find your solution valuable, they may be willing to pay more for the product. Solutions that enable customers to generate revenue can often be more in-demand for customers than solutions that support internal, non-revenue-generating functions. However, solutions that improve the efficiency and effectiveness of their customers' operations can still be successful, as we see in the following example scenarios.

This page and the wider journey are meant to provide ISVs with step-by-step guidance to ideate, develop, and publish new applications. However, there are other Microsoft resources available that can help you drive organizational buy-in around your use case. The [Microsoft AI Value Accelerator \(MAIVA\) Playbook](#) provides foundational guidance to enterprise customers for industrializing AI. This resource can help you align your use case with overall business goals, standardize your AI strategy, and effectively scale the value your solution provides.

Costs

Accurately determining the costs of developing and operating a generative AI application is challenging because of the diversity of variables at play. Individual applications involve different tools and architectures in their development, creating a wide variety of possible initial and ongoing costs for solutions. However, there are tools that can help you estimate these costs. You can use the [Azure pricing calculator](#) to input your own information and receive a customized pricing estimate online. You can also find out more about costs associated with specific development tools such as the [services within Azure AI Studio](#) and [Copilot Studio](#).

Generally, as you assess the costs involved with your application, there are multiple elements to plan for. Generative AI models typically require up to six primary components:

- **Database or storage:** Managing costs for storing your application's data is crucial, especially when handling both structured and unstructured data across multiple locations to ensure availability and redundancy.
- **Index and search engine:** Investing in an efficient index and search engine is essential to making your data accessible and useful, and enabling users to quickly retrieve the information they need.
- **Model development and training:** Budgeting for the development, training, and refinement of your AI or machine learning models is critical, as these models form the foundation of your application and influence its effectiveness.
- **Application hosting environment:** Selecting a cost-effective hosting environment, whether it's Azure Kubernetes Service (AKS), Virtual Machines (VMs), or AppService, is vital, as it directly impacts your application's performance, scalability, and security.
- **Cache:** Implementing caching solutions, though technically optional, can lower operational costs by reducing database strain and improving response times.
- **Potential security or monitoring service:** Prioritizing expenditures on security and monitoring services is crucial to safeguarding your application. It helps ensure

compliance, and maintain optimal performance, preventing costly disruptions or breaches.

It's important to remember that your application development process and internal functions can heavily impact costs. Thinking about your application holistically and considering all relevant factors can help drive a successful commercial strategy.

Go-to-market (GTM) approaches

After considering the value your solution provides, along with the costs involved with its development, deployment, and operation, you're ready to determine the approach that fits your scenario. Through our work with ISVs, we've identified four primary go-to-market approaches for generative AI applications. While not mutually exclusive or comprehensive, these approaches provide a tangible representation of strategies successfully applied by ISVs.

- **Free** - the solution is offered to end-users freely, either through a stand-alone download, or as an update to an existing free solution.
- **Freemium** - the solution includes a free base product with extra, widely applicable features through a paid premium version or license.
- **Paid Feature** - the solution is a separate feature on a larger platform or solution with its own charge but is narrowly applicable to a select user base with specific needs.
- **New Solution** - the solution offers clear value to the customer and reaches the level of sophistication that warrants it being monetized and sold as a separate, new solution. This solution can potentially serve as a platform for future offerings delivered through the other approaches.

When choosing your go-to-market strategy, it's important to weigh the cost and benefits of each approach against the investment required and the value of your desired outcome. For example, if you're thinking about developing a free solution, you need to fully consider the solution's purpose and benefits for it to pay off. To that end, our experience with ISVs suggests that aligning to one of the following approaches can help drive a successful and consistent commercial strategy. Here is a summary of these approaches:

	Free	Freemium	Paid Feature	New Solution
 Goal of your solution	Offer a free new feature to generate market share and name recognition	Offer a free base product with additional features through a paid premium version or license	Provide a specialized solution feature at a premium	Offer a net new solution or platform to existing or new customer base, generate additional revenue
 Target Audience	Customers who are interested in ISV's core solution but are interested in AI automation of the workload	A range of users with different use cases including some willing to pay for advanced features	A select user base willing to pay an additional charge to use a specific feature that meets an unmet need	Customers willing to invest in a new comprehensive generative AI solution
 Intended value for the end-user	Enhanced productivity and/or efficiency tied back to ISVs core solution	Enhance productivity and/or efficiency at scale	Effectiveness and/or Efficiency – giving customers a chance to do something that was costly or cost prohibitive	Sophisticated AI features capable of addressing new or underserved customer needs
 ISV investment	Low – higher value seen in the ISVs core solution	Low for free product, additional investment allocated based on paid feature consumption	Medium – This becomes a core feature in future sales opportunities	High – disruptive case where generative AI becomes the product

To explain these scenarios, we examine several examples of applications using these go-to-market approaches, including how they selected their commercial strategy, and the benefits they experienced from using this approach. To thoroughly examine each of these case studies, we use the set of considerations we discussed earlier:

- **End user** - the personas that use the solution
- **Business value** - how the solution provides value to customers
- **Technical approach** - how the solution is built
- **Costs** - the initial and ongoing costs of running the solution

Let's look at how real-world ISVs successfully put these go-to-market approaches into action for their AI and generative AI solutions.

Commercialization example scenarios

ISVs can adopt various commercial strategies for their solutions, depending on their product's strategic goals. Based on our experiences with ISVs, we've put together four generalized scenarios to explain each GTM strategy. For the purposes of this article, we refer to our example ISV as Contoso, a fictional software developer.

Free scenario: Invoice management copilot feature

Contoso, seeking to expand the reach of their data management software, conducted a survey to identify a need they could solve to attract their target audience. They learned that many of their executive customers were frustrated with the amount of time lost to convoluted administrative processes like invoice approvals. From an initial email

notification, it could take up to a few days for decision makers to just approve an invoice for payment. This involved locating the required information to confirm across multiple departments if contract terms were fulfilled.

To drive demand for their core application among potential customers, Contoso built a copilot feature that helps improve efficiency within the customers' administrative operations. They designed this copilot feature to help reduce time that decision makers spend approving invoices for payments and focuses on the interface where users collaborate the most: Microsoft Teams. Contoso created this copilot to complement its larger data management program and showcase the full capabilities of the generative AI solution. Contoso offered this copilot feature from their overall solution to users for free to enhance the value for their target audience and drive demand towards the larger platform. They plan to sell the larger solution as a fully SaaS (Software-as-a-Service) based offering, once the user base increases.

Let's examine why Contoso chose this commercial approach:

- **End users:** Decision makers seeking operational improvements through more efficient administrative operations. Contoso is specifically targeting users interested in joining their larger data management platform for its productivity benefits.
- **Business value:** The copilot feature improves customer efficiency by quickly finding contract data and contacts relevant to each invoice. It also saves time by simplifying collaboration across departments to verify milestone completion status within the Teams UX.
- **Technical approach:** Contoso extended some capabilities from the core application using Pro-Code and Teams Message Extensions, Teams Copilot for UX, and Azure SQL database.
- **Costs:** Broadly low building and testing costs including development resources and cloud services costs but applying investment in core solution.

With this copilot feature offered at no cost, executive decision-makers can access actionable information easily in Teams Copilot, streamlining the approval process and helping improve efficiency without a significant monetary investment.

By bundling the Teams Copilot capability as a part of the larger Copilot solution, Contoso:

- Showcased their software's capabilities to the target user base.
- Enhanced the value of the core Gen AI solution for a new set of target users, leading to greater success later for the overall solution.

Releasing this copilot for free enabled Contoso to provide an efficient solution to their users' frustrations and bring their software and brand to executives with purchasing power. This approach empowered Contoso to position themselves as leaders in the AI space, building brand reputation and boosting market share.

Freemium scenario: AI-enabled photo editing features

After identifying a need for faster and easier photo editing software, Contoso developed image editing software that uses AI to enhance professional level photographs in addition to providing other standard editing features. The solution is marketed to individuals and enterprise organizations as a SaaS platform, enabling photography professionals to quickly adjust photos whether they work on their own or in a group.

Customers can either subscribe to a free base version of the application, which includes a daily limit on certain features and fewer capabilities, or a premium version. The premium version provides users with more complex capabilities, such as adding in objects or changing poses in photos, and removes usage limits.

Let's examine why Contoso chose this commercial approach:

- **End user:** Contoso needed their application to be widely accessible to a wide range of individuals and groups. Though their software was designed with photography professionals in mind, they knew hobbyists and student groups would also make up a large percentage of end users.
- **Business value:** The solution's AI features hasten the photo editing process, saving customers time, and increasing efficiency for application users. The premium version also includes capabilities to improve the effectiveness of the underlying platform, increasing the product's value to customers.
- **Technical approach:** Contoso developed the full platform using elements of Microsoft Fabric and Azure AI Studio. While the larger platform demands continuous support, the free version was developed using low code and required minimal development lift.
- **Costs:** While designing the app, Contoso considered costs for GPU utilization, development, and continued application maintenance. They also noted a high storage cost because their application requires a large model and quantity of data.

Given their application's wide applicability and breadth of end users, Contoso knew they would benefit from a large-scale, highly accessible platform. By providing their feature for free to a large customer base, Contoso increased productivity and reduced costs for a larger number of customers. Offering a free option also made their application accessible to more customers, which greatly increased name recognition and the

popularity of their product. Because of the free version's popularity and it allowing users to experience some of the solutions benefits, more users subscribed to the paid model.

During the first few months of their application's deployment, Contoso:

- Enhanced brand recognition and market presence by providing a free option available to a wide range of customers.
- Increased revenue by driving adoption of the paid subscription among professional and enterprise customers.
- Positioned their application as a clear choice for users and industry standard by being accessible to both professionals and students.

Choosing a freemium approach enabled Contoso to expand its customer base quickly and draw in new users. Despite high costs to develop and maintain their solution, Contoso quickly drew in many paying subscribers, which increased overall revenue.

Paid feature scenario: AI industry marketing assistant

Contoso offers a comprehensive marketing resource management platform that makes it easy for companies to organize their digital assets and create impactful, omnichannel experiences. While the platform already offers a selection of core marketing automation and digital asset management capabilities, Contoso noticed an opportunity. They could help their customers optimize marketing efforts at scale by embedding a new generative AI-powered, copilot-like assistant directly in their platform.

In addition to a generic assistant, Contoso developed a suite of industry-tailored assistants for customers with specific needs. These industry marketing assistants have specialized knowledge and capabilities to support industry-specific functions. They are available to customers at an additional charge over the generic assistant.

Let's examine why Contoso chose this commercial approach:

- **End users:** Contoso wanted to target power users who are interested in applying new technologies to optimize their workflows. They also wanted to broaden their market share by offering next-gen, industry-leading features on their core platform.
- **Business value:** The copilot helps increase customer effectiveness and efficiency by making it easy for users to quickly create high-quality content to meet their industry's needs.
- **Technical approach:** Contoso developed the copilot with Azure OpenAI Assistants and Azure AI Studio, investing in top-line models like GPT-4.0 to help drive complex and accurate assistant engagement.

- **Costs:** The copilot's costs included development costs, ongoing cloud service charges to run the application in production, and Azure OpenAI subscription costs, warranted by the feature's demand in the market.

Offering these industry-tailored copilots as a paid feature enabled Contoso to appeal to specific audiences within existing and net-new customers. At the same time, Contoso limited costs and development times by using a low-code approach, ensuring their investment aligned with the revenue generated by the feature.

In doing so, Contoso:

- Increased their market share by enhancing the value of their existing platform and driving adoption among new customers.
- Unlocked new revenue streams from power users that want to experiment with new technologies without alienating their core customer base.
- Implemented a quick-win, generative AI feature without incurring significant development costs.

Adding this paid feature to their existing platform enabled Contoso to experiment with a new generative AI capability without incurring high costs or alienating core components of their customer base. This approach helped them establish a solid generative AI foundation on which to expand with future services and features in the future.

New solution scenario: Contact center agents

Contoso identified an opportunity for increased AI automation in contact centers. Many customer service teams were overwhelmed by the volume of calls, but Contoso realized that many contact requests were for simple tasks or information that an AI assistant could provide. These AI agents could alleviate the loads on human representatives, enabling them to deal with the most complex issues, dramatically increase the efficiency of closed tickets, and reduce overall costs for the contact center.

To meet this demand, Contoso developed a generative AI contact center agent that can answer questions, initiate tasks, and close tickets for customers over the phone or through a text-based interface. Given the market demand and disruption potential of this technology, Contoso built this application as a new, SaaS-based solution, charging a base subscription fee along with a small additional usage charge. Let's take a look at the factors that went into

Contoso developing their application with this approach:

- **End user:** This application would be targeted towards customers with queries or issues that would contact companies' call centers. The application's UI, in the case of chat interfaces, prioritizes simplicity to enable ease of use by the customer.
- **Business value:** This solution improves both the efficiency and effectiveness of contact centers by reducing the cost-per-ticket-closed and empowering contact center workers with further capabilities and support.
- **Technical approach:** Contoso built the solution using pro-code development with Microsoft Fabric, Azure AI Studio, and Azure OpenAI Assistants. The value this application provides warranted the substantial development lift and continuous support.
- **Costs:** Contoso's costs included development resources, cloud-service costs during development, testing, and production, and continuous support, justified by the core generative AI feature.

Offering their solution as a SaaS model enabled Contoso to manage their application's model usage to flexibly meet demand while simultaneously layering on top of existing legacy contact center systems. These AI agents can intake data from various sources and pass it through multiple levels of automated decision-making to rapidly identify and address operational tasks. As users become more accustomed to engaging with AI-powered interfaces, the underlying core system can become commoditized-valued more as data servers than core systems. This change offers a major potential for disruption in the industry, justifying Contoso's decision to develop their application as a new solution.

With this solution, Contoso:

- Deployed another revenue opportunity through a core solution platform that improves customer efficiency.
- Developed a new, disruptive interface for contact centers that transforms how customers engage with customer service.
- Positioned growth opportunities through future agents, features, and SaaS-based capabilities that can be offered through the platform.

Carefully considering the elements of their solution helped Contoso craft a commercial approach that aligned with their goals and application platform as a whole. The SaaS model that Contoso used for this solution is widely applicable and offers a transformative component to the software that ISVs develop.

Learn more about SaaS

While developing your solution in a SaaS format can present as a larger initial lift, it ultimately enables greater simplicity and scalability with customers. It may seem simpler to avoid building a SaaS solution from the ground up, but adopting an on-premises or

packaged software approach can quickly become costly and inefficient as your customer base expands. Resources and guidance from Microsoft can help enable your SaaS journey. For more information on developing a SaaS solution, see the following articles:

- [Foundations of SaaS](#) - Microsoft Learn training focused on the characteristics and benefits of a SaaS model
- [Overview of the journey for designing SaaS and multitenant solutions](#) - Holistic documentation on the overall journey of developing a SaaS solution
- [Starter web app for SaaS development](#) - Initial architecture guidance for SaaS applications

Publishing and sales enablement

Once you establish your commercial strategy and go-to-market approach, your next step is to choose where you want to publish your solution for customers. Microsoft offers two distinct storefronts that allow partners to list offers, enable trials, and transact directly with customers and the Microsoft ecosystem: [Azure Marketplace](#) and [AppSource](#). Azure Marketplace offers a more technical and infrastructurally based storefront through which to engage with customers. You can find comprehensive [guidance around publishing on Azure Marketplace here](#). The AppSource storefront offers a place for applications oriented towards business decision makers, including Dynamics 365 and Power Platform applications. You can find [guidance for publishing on AppSource here](#). These storefronts are differentiated by target audience and product to help customers quickly find what they need.

[+] [Expand table](#)

	Azure Marketplace	AppSource
Target audience	IT professionals and developers	Business decision-makers
Built to extend	Azure	Azure, Dynamics 365, Office 365, Power BI, Power Apps
Types of solutions	Infrastructure solutions and IT-focused professional services	Finished line of business apps and professional services
Publishing options	Contact me, consulting services offer, trial, virtual machine, solution templates, and managed apps	Contact me, consulting services offer, and trial
In-app experience	Azure portal and CLI	Office 365, Dynamics 365, Power BI, Office client apps

Once you've developed your application and aligned on your approach to commercialization, consider how to align your publication strategy with your solution to drive maximum adoption and impact.

Conclusion

As you consider the commercial strategy for your solution, it's important to remember that there is no one-size-fits-all approach. Holistically considering each of the different factors at play—from potential value and costs, to go-to-market approach—is critical as you determine the best way to monetize your product. In addition to your overall commercial strategy, it's also important to consider how your product is priced. You can find more information on [pricing models for a multitenant solution here](#).

Another resource that can support you as you consider and develop your application's commercial strategy is the [ISV App Advisor](#). This self-guided experience can surface the latest resources and recommendations based on your current development stage.

Understanding and planning your commercial strategy is a key step towards success as you [evaluate and prioritize a use case](#) and determine the best approach to [developing your solution](#). Across the ISV Journey, aligning your application's development and positioning with your commercial strategy helps drive a successful deployment and product offering.

Once you establish your use case, how you plan to build it, and your commercial strategy, the next step is determining the specific [low-code](#) or [pro-code](#) pattern to follow as you develop your solution.

Feedback

Was this page helpful?

 Yes

 No

Evaluate and Prioritize an AI Use Case with Business Envisioning

Article • 09/20/2024

- AI is transforming how industries operate
- Business Envisioning
 - Identify potential use cases
 - Example scenarios
 - Business, experience, technology framework
 - Evaluate and prioritize use cases
- Next Steps

Introduction

Artificial intelligence (AI) is revolutionizing industries, prompting businesses to embrace its potential like never before. It has the potential to unlock new opportunities and enable transformative use cases, reshaping the way businesses interact with and provide products and services to their customers. As AI becomes more common in daily applications, we need to ensure that AI projects match business goals, meet user needs, and can be built with the resources we have.

This article enables Independent Software Vendors (ISVs) to:

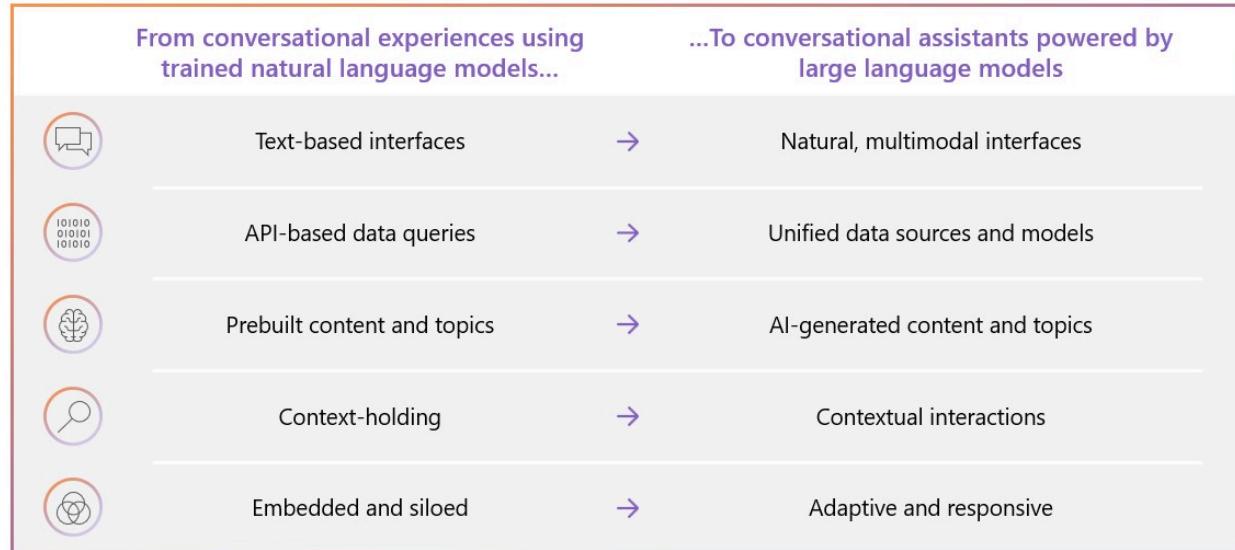
- Learn how to identify and refine potential use cases
- Evaluate the viability of use cases
- Understand how to approach use case prioritization

AI is Transforming How Industries Operate

AI is breaking down barriers across industries, from enhancing manufacturing productivity to minimizing the environmental impact of supply chains. At Microsoft, we believe in providing you with the tools to build the best solution you can, whether that means using traditional AI and machine learning (ML) or cutting-edge generative AI (GenAI) models.

Our partners are increasingly looking to develop new solutions and features that apply this advanced technology and deliver greater value to customers. Many of our partners are focused on the potential of generative AI and its potential to transform the way work is being done. While traditional AI and ML certainly have their place, GenAI offers a

step forward in the progression of artificial intelligence. While traditional AI and chatbots existed for some time, generative AI marks a significant advancement in its capabilities and potential.



The possibilities of this transformational technology means that effectively applying the value of AI can be a major undertaking, even for large companies. While this page, and subsequent ones, are designed to help ISVs identify, build, and deploy a viable generative AI use case, there are more resources available. We developed the Microsoft AI Value Accelerator (MAIVA) Playbook to support enterprise customers with foundational knowledge and guidance for enabling AI at scale across their organization. Though not designed as a tactical guide for ISVs, MAIVA can provide valuable insights into the diverse perspectives of the different roles involved in bringing a GenAI application to life. This [open-source framework](#) helps you understand what makes deployment and operation successful, why it's challenging, and offers advice on leadership, building a data-driven culture, and managing change.

While this development in AI technology brings huge potential for new applications across industries, each opportunity should still be evaluated for its viability. The remainder of this page is devoted to helping ISV business decision makers identify and evaluate the potential business value, user demand, and technical feasibility of proposed AI use cases.

Business Envisioning

Identify Potential Use Cases

This page provides a consistent and comprehensive framework to help you evaluate the viability of potential AI and GenAI use cases. Ideas for use cases can come from

anywhere, but starting with brainstorming, researching use cases online, and reviewing [Microsoft customer success stories](#) can help get the process started.

Microsoft possesses a deep well of experience in this space and collaborates with partners across industries to develop innovative AI use cases. Here are some examples from across industries:

[+] [Expand table](#)

Industry	Use Case	Description and Benefits
Manufacturing	Factory assistant	Helps factory workers quickly identify equipment issues, find solutions, train and improve skills, and generate Programmable Logic Controller (PLC) code using natural language with translation features.
Healthcare	Claims management	Offers a conversational AI assistant experience that can help claims agents complete authorization and administrative tasks quickly. It is also capable of creating comprehensive authorization strategies based on claims data and improving visibility into the status of claims and authorizations.
Financial services	Corporate banking	Helps bankers forecast client needs based on historical data and interactions, automate routine administrative tasks, and monitor client flows in real time.
Sustainability	Supply chain characterization	Enables sustainability practitioners to review supplier data to ensure compliance with environmental standards, track product lifecycles, and optimize supply chain operations.
Retail	Store operations assistant	Empowers store employees to query operating policies and procedures, accelerate onboarding and upskilling, and review FAQs through an interactive conversational assistant.

As you develop use cases, it can be helpful to consider the following questions:

- **Problem:** What is the problem to be solved? What are the underlying root causes? How does the current process work?
- **Opportunity:** What is your idea to capture the opportunity or solve the problem? Why do you believe this idea is a good approach?
- **Business objective:** What business objective is being achieved by enabling this use case?
- **Measurement of success:** How do you define success? How is success measured? What are the measurable key results that show progress?
- **Accountability:** Who is the executive sponsor for the solution? Who is accountable for the Objectives and Key Results (OKRs)?

You can probably come up with different AI solutions to address various use cases. Business envisioning helps you evaluate and prioritize the most viable use cases for development and execution. Start with a description of the problem and your use case. Analyze factors such as the business objective, key results, and primary stakeholders to determine a strategic fit score for your application, 1 to 5. Business envisioning can serve as your starting point for bringing these various use cases to stakeholders who are needed for development and execution.

Business envisioning use case template

The screenshot shows a template for business envisioning. At the top, there are fields for 'Use case name:' and 'Department:', and a dropdown for 'Internal or external use:'. Below this is a 'General' section with two main fields: 'Problem to solve' (describing the challenge and current solution) and 'Use case description' (what the idea is). Underneath is a section titled 'Objectives & key results – Strategic fit' with four columns: 'Business objective' (describe the business objective to be achieved), 'Key results' (3-5 measurable Key Results), 'Primary stakeholder who is accountable for OKRs' (Role, Name, Org/Group), and 'Strategic fit' (a scale from 1 to 5). A note at the bottom indicates '1 = Low and 5 = High'.

In the following section, we review how our ISV Contoso used this framework to assign two different use cases strategic scores when we do so for our example scenarios. For now, set this strategic fit score aside. We'll return to it when we discuss use case prioritization, after evaluating the business value, user experience, and technical feasibility of your use case.

Example Scenarios

To help guide you through this business envisioning process, we'll review two potential use cases that ISV Contoso analyzed with their customer AdventureWorks, a large retailer. We examine each use case in-depth and compare them to determine which presents greater value and viability for development.

Use Case 1: Contoso Store Operations Assistant

In this scenario, AdventureWorks was struggling with the following problems:

- High rate of employee turnover
- Slow and inefficient employee onboarding

- Low productivity of frontline workers
- Store managers lack effective tools to manage operations

To address these issues, Contoso and AdventureWorks developed a Store Operations AI assistant use case. This assistant would be capable of providing employees with guidance on onboarding materials, store policies and procedures, and task creation. To begin their evaluation, Contoso calculated the strategic fit of this use case using the three factors discussed: business objective, key results, and primary stakeholders.

- **Business objective:** Contoso aims to empower AdventureWorks to quickly onboard new employees, improve employee efficiency, and increase store productivity.
- **Key results:** The Store Operations assistant targets improved Net Promoter Score, increased employee retention, and adherence to compliance and policy requirements.
- **Primary stakeholders:** Contoso identified engineering and product leads to collaborate with their development team on this application.

Based on these factors, Contoso assigned this use case a strategic fit of 4 given that its business impact was not necessarily a revenue-generating opportunity. This high score suggests that there is a strong fit for this use case to help AdventureWorks with the current difficulties they're facing.

Use Case 2: Contoso Shopping Application

In this scenario, Contoso worked with AdventureWorks to identify the following key business and technical problems to address with this use case:

- A lack of a digital storefront
- Constrained employee communication
- Difficulties in planning for inventory allocation
- No automated way of tracking insights such as popular items and shopper behavior

To address the situation, Contoso proposed developing the Contoso Shopping application. A comprehensive solution capable of supporting a digital storefront, connecting employees, and tracking and analyzing data. Using these components, Contoso evaluated the strategic fit of this use case.

- **Business objective:** Contoso sought to solve AdventureWorks struggles with a lack of a digital storefront, deploying an application capable of supporting customer transactions, inventory management, and comprehensive communications.
- **Key results:** Contoso and AdventureWorks identified the successful deployment of this application, specifically focusing on many end-users per day and improved

revenue as key performance indicators, as their primary key result.

- **Primary stakeholders:** Contoso has a full development team assigned to the project, with extra subject matter expertise from AdventureWorks ready to go.

Based on these factors, Contoso assigned their use case a strategic fit of 5. This high score suggests a strong fit for Contoso and AdventureWorks and brings us to the next step in the business envisioning process.

After examining the details of each use case and assigning them a strategic fit score, let's evaluate how they measure up in terms of business value, user experience and demand, and technical viability using the business, experience, technology (BXT) framework.

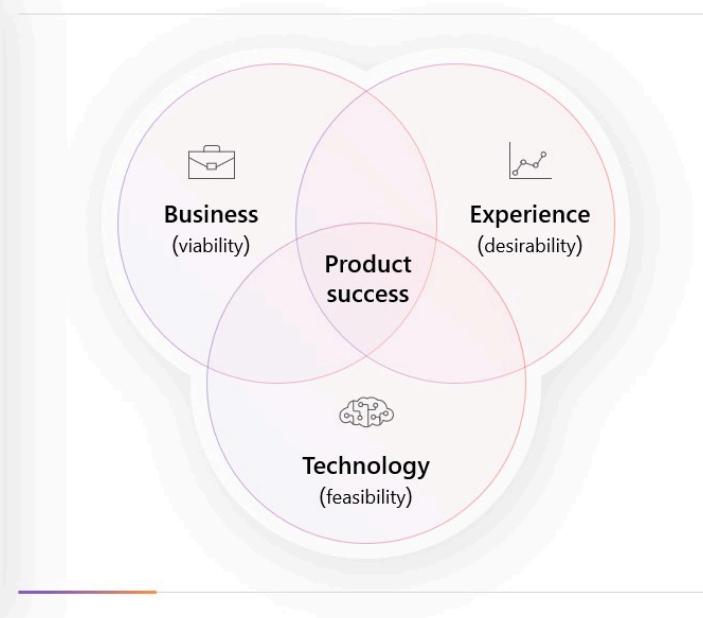
Apply the Business, Experience, Technology Framework

The Business, Experience, Technology (BXT) framework enables ISVs to evaluate the potential of their use cases. This exercise is designed to comprehensively walk through the details of each of your use cases and evaluate their viability. This is where you can introduce your use cases to the wider ecosystem of stakeholders who might be involved in their development, from security and compliance experts to technical development teams. Applying the BXT with a wider group not only helps drive clarity on the realistic implementation of each use case, it also helps ensure buy-in from stakeholders whose effort will be required to make this vision a reality. The high-level process includes the following:

- Three high-level focus-areas: business viability, experience and desirability, and technological feasibility.
- Within each of these areas, there are three subcomponents, each with their own score.
- Following this, you'll rank and visualize the viability of your use cases using a combination of these components.

We walk you through this framework step-by-step using the two example use cases we introduced earlier: Contoso store management assistant and Contoso Shopping application.

The BXT framework



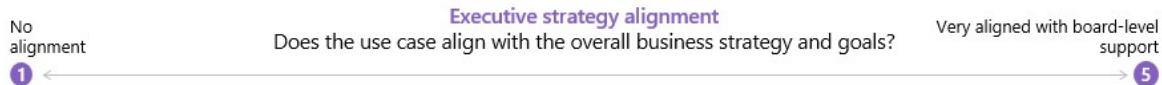
Business

First, it's important to ensure that the prioritized use case aligns with your overall strategy and has real opportunities for revenue. The primary consideration under Business is the financial viability of the solution. To fully answer this question, we've broken this part of the framework down into three components: executive strategy alignment, business value, and change management timeframe.

If you'd like to dig deeper into how to drive internal alignment around your use cases and the value they provide, the [Microsoft AI Value Accelerator \(MAIVA\) framework](#) can help. This framework provides comprehensive guidance including AI strategy planning, experimentation cycles, data platforms, and deployment.

Executive Strategy Alignment

Your selected use case should align with your organization's overall mission and teams. For a use case to be successfully developed, key stakeholders and impacted users should be clearly identified and ready to engage on the effort. This helps avoid inefficiencies by aligning teams, resources, objectives, and development efforts and avoiding redundant or nonproductive work.



Business Value

A successful use case is one that generates value for the business. While this can often take the form of generating value for end-users, and thus revenue opportunities, value can also look like improved productivity and cost-effectiveness for the customer internally. Regardless of the value generated, one of the major things to consider is your approach to monetization; for example, will this be a core product or a free add-on? Four primary approaches have begun to materialize on this front:

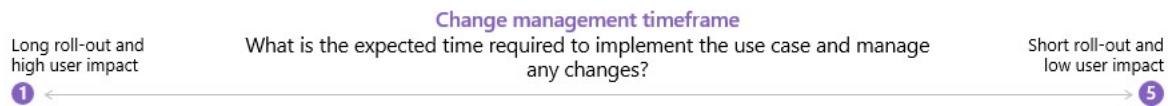
- **Core:** the application is mission critical for core product experience, a must-have.
- **Extend:** the application enhances product experience but may not be mission critical for all users, a nice-to-have.
- **Add-on:** the application adds significant value to a particular subset of users for the present.



Considerations around commercial strategy and monetization are critical to determining the business value of your application. You can learn more about what goes into this decision on the Commercial Strategy and Monetization for ISV Applications page, coming soon.

Change Management Timeframe

The development timeline is an important factor in the business viability of your use case. Key milestones to consider are when your application will be ready for internal testing, when it is ready for initial pilot users and when it can begin generating revenue. Inaccurate time estimates could lead to costly and inefficient delays, which can after impact overall business health and pull valuable resources away from more effective or productive efforts. In addition, the implementation and maintenance of product changes can extend beyond release. Use cases that require less fundamental change and have lower impact on operations can serve as achievable quick wins. To help mitigate the potential drawbacks of complex development projects, it's critical to be realistic with how you approach timeline estimates.



It's important to note that a more complex use case with a longer roll-out and higher user-impact is not necessarily an unviable use case but will likely require more resources and time to develop and deploy, and potentially introduce new obstacles. For example, an application with a longer roll-out potentially has a longer path to profitability. This

does not preclude the application from development, but it's important that ISVs make these decisions with eyes wide open, and balance considerations across the BXT to ensure the intended use case fits their needs, circumstances, and available resources.

Example Use Cases Business Evaluation

Let's examine how the example Contoso use cases we introduced land with each of these subcomponents and assign them a score. We use each of these scores, together with the strategic fit score assigned earlier, to prioritize which use case to develop. You'll score each subcomponent separately, and then take the average of those three scores to find this component's overall score.

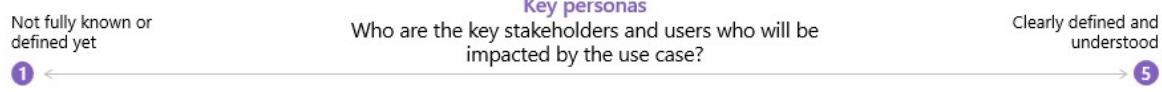
Business viability – Strategic business impact assessment			
Store operations assistant	How does the use case align to the organization's executive strategy?	How does the use case generate business value?	Business impact
	<ul style="list-style-type: none">Modernizes employee experiences and toolsDrives improved productivity and efficiencyAligns with AdventureWorks' executive strategy	<ul style="list-style-type: none">Reduces task completion time and errorsOptimizes staff resources to help improve worker and store productivity	<ul style="list-style-type: none">Presents a minor change to employees' experienceDeploys over a short roll-out periodCreates generally minimal impact
Shopping application	Score: 3	Score: 2	Score: 4
	How does the use case align to the organization's executive strategy?	How does the use case generate business value?	Business impact
Shopping application	<ul style="list-style-type: none">Modernizes employee experiences and toolsDrives improved productivity and efficiencyAligns with AdventureWorks' executive strategy	<ul style="list-style-type: none">Reduces task completion time and errorsOptimizes staff resources to help improve worker and store productivity	<ul style="list-style-type: none">Presents a minor change to employees' experienceDeploys over a short roll-out periodCreates generally minimal impact
	Score: 5	Score: 5	Score: 3
			Overall Score: 3
			Overall Score: 4.3

Experience

Once you've identified a use case's business value, you need to ensure there is demand for the solution. The experience component of the BXT framework focuses on the needs and preferences of the user, and if the solution and experience is desirable to them. To best answer these questions, we've broken down Experience into key personas, value proposition, and change resistance.

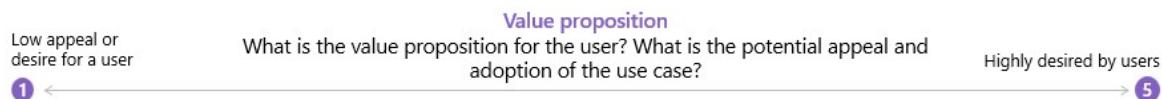
Key Personas

Understanding the key personas relevant to your use cases is critical to identifying existing demand. This includes both ultimate end-users but also stakeholders involved in building and maintaining the application. Having internal resources prepared and committed will help drive an efficient development process. Understanding these personas enables developers to prioritize use cases that address the specific needs, challenges, and goals of these critical groups, ensuring the application delivers maximum impact and value to those who matter most.



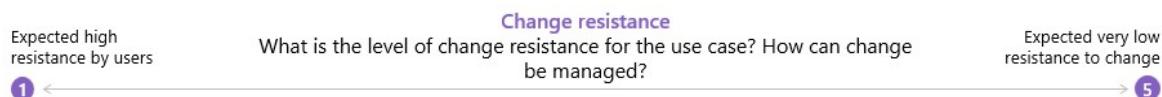
Value Proposition

The value proposition of a use case outlines the benefits and advantages that users gain from using the application. This includes aspects like increased efficiency, cost savings, improved productivity, or enhanced end-user experience. Understanding the value proposition is vital as it determines the potential appeal and adoption of the use case. A compelling value proposition can help ensure that the application not only meets end-user needs but also stands out in a competitive market, enhancing revenue opportunities.



Change Resistance

Change resistance is another factor to consider when prioritizing use cases as it reflects the users' willingness to adopt new technologies and processes. High levels of change resistance can hinder the successful implementation and adoption of an application. Understanding the level of change resistance enables developers to devise strategies to manage and mitigate resistance, such as enhanced training and usage-guidance. Addressing change resistance effectively can help ensure smoother transitions and higher acceptance rates, leading to more successful and sustained application usage.



Example Use Cases Experience Evaluation

Let's examine how our example uses cases align with each of these subcomponents.

Experience value – User desirability impact assessment

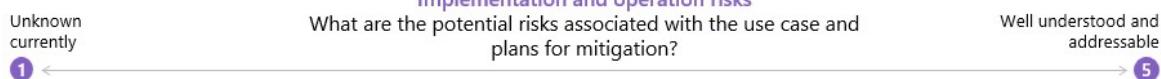
Store operations assistant	Who are the key personas?	How appealing is the overall value proposition for the user?	What is the change resistance?	User desirability
	<ul style="list-style-type: none"> Will directly impact: <ul style="list-style-type: none"> Store associates Managers Demand planners 	<ul style="list-style-type: none"> Improves productivity for managers Enhances onboarding experience for employees 	<ul style="list-style-type: none"> Presents low change resistance given the prevalence of natural language assistants 	This use case doesn't reach consumers directly, and presents minor value, but presents low potential change resistance.
	Score: 4	Score: 3	Score: 4	Overall Score: 3.7
Shopping application	Who are the key personas?	How appealing is the overall value proposition for the user?	What is the change resistance?	User desirability
	<ul style="list-style-type: none"> Improves access for customers Empowers store managers and corporate teams with new data-driven capabilities 	<ul style="list-style-type: none"> Provides huge value to AdventureWorks by expanding digital access Expands market share and revenue opportunities 	<ul style="list-style-type: none"> Creates a new system and interface for employees to learn to use Presenting a medium to large potential for change resistance 	This application exponentially increases access to AdventureWorks, presenting enormous value, but the solution's complexity presents potential change resistance.
	Score: 4	Score: 5	Score: 3	Overall Score: 4

Technology

Lastly, when prioritizing use cases, considering technical factors such as technical feasibility, integration challenges, and the accuracy of AI models is essential for the ultimate efficacy of the application, and the development process. To comprehensively review the technical feasibility of the use case, consider the following three topics: implementation and operation risks, sufficient safeguards, and the fit of AI and Large Language Models (LLMs).

Implementation and Operations Risks

Identifying and taking steps to mitigate implementation and operational risks can help developers avoid delays, bugs, and cost overruns, ultimately helping to drive the success of the application. Potential risks can include technical issues, resource constraints, and data security concerns. Mitigation strategies can involve thorough testing, contingency planning, and ensuring robust support and maintenance strategies. By addressing risks early, developers can prevent disruptions and ensure a smoother implementation process, leading to more reliable and sustainable application development and deployment.



Sufficient Safeguards

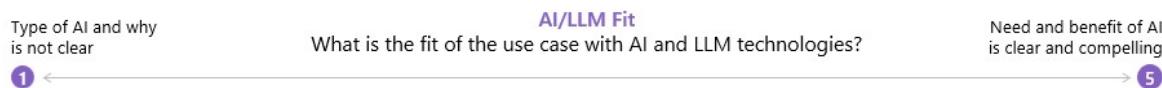
Managing risk and compliance, especially with regards to AI, is crucial to the successful development and deployment of your application. Proactive measures such as robust security and compliance measures, data protection and access controls, and responsible and safe AI standards, can help protect users and their data. This also includes ensuring

compliance with all relevant legal and regulatory requirements. By prioritizing use cases with well-established safeguards, developers can enhance the reliability and security of their applications, fostering greater user confidence and minimizing potential liabilities.



AI/LLM Fit

In order to effectively implement AI into a use case, the fit of AI and LLMs with the use case must be evaluated. Use cases that align well with AI and LLM capabilities can benefit from enhanced automation, improved decision-making, and personalized user experiences. Understanding how these technologies can be integrated effectively helps maximize their potential and address specific user needs more efficiently. By prioritizing use cases that are well-suited for AI and LLM implementation, developers can create innovative applications that offer significant competitive advantages and drive substantial value for users.



Example Use Cases Technology Evaluation

Lastly, let's look at how our example uses cases align with each of the technology subcomponents.

Technical value – Feasibility impact assessment				
Store operations assistant	What are the implementation and operational risks?	Are sufficient safeguards available?	How is the AI/LLM fit?	Technical feasibility
	Presents potentially minimal operational risks such as: <ul style="list-style-type: none"> Employees not using or adopting the solution Integration difficulties integrating existing systems 	Includes safeguards such as: <ul style="list-style-type: none"> Access controls Security audits Data governance strategy 	<ul style="list-style-type: none"> Converses in natural language Responds to a variety of queries with answers based on customer data. 	The simplicity of the store operations assistant makes it very feasible with low risks. Overall Score: 4.3
Shopping application	What are the implementation and operational risks?	Are sufficient safeguards available?	How is the AI/LLM fit?	Technical feasibility
	<ul style="list-style-type: none"> Presents greater implementation risks given its complexity Contoso's development experience helps mitigate this risk 	Includes: <ul style="list-style-type: none"> Built-in regulatory compliance Data access controls and governance strategies Comprehensive security features 	<ul style="list-style-type: none"> Leverages AI and GenAI in multiple ways including: Conversational assistants Machine learning-based data analytics 	The complexity of the application presents greater risk, but Contoso's expertise and the AI fit present huge opportunity. Overall Score: 4.3

Now that you've considered each of these factors regarding the viability of your use cases, how do we evaluate the results?

Evaluate the Viability of Your Use Case

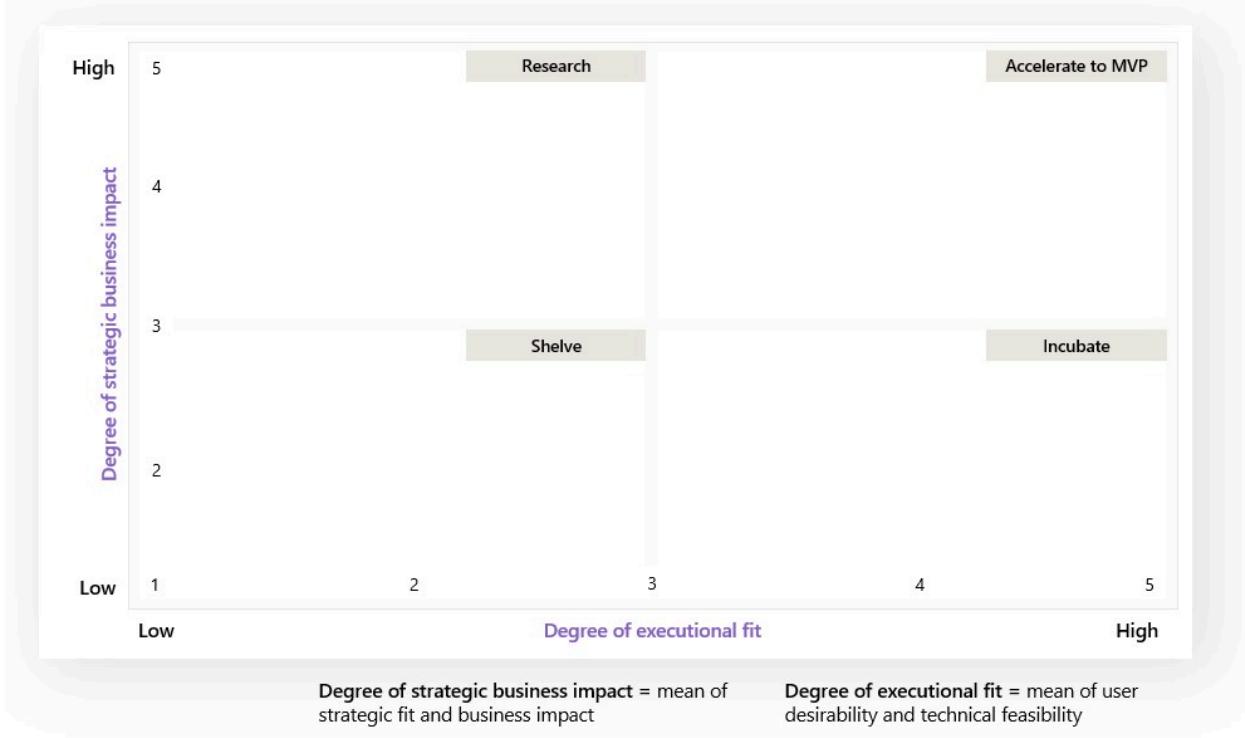
Once you've examined your use cases using the BXT framework, the next step is to evaluate which use case to prioritize for development. At this stage, each use case you're examining should have three scores, one for business, experience, and technology each. You can use this template to consolidate the BXT and subsequent scores for your use cases:

Use case name: Department:		Internal or external use:					
Business viability – Strategic business impact assessment							
How does the use case align to the organization's executive strategy? How does the use case contribute to your organization's executive roadmap / strategy?	How does the use case generate business value? e.g., additional revenue though a new service or cost saving through higher efficiency	Business change management timeframe Estimate based on people, tools and processes impact. <input type="checkbox"/> 1 - 3 months <input type="checkbox"/> 4 - 6 months <input type="checkbox"/> >6 months	Business impact 1 2 3 4 5 1 = Low and 5 = High				
Experience value – User desirability impact assessment							
Key persons? Who are the users benefiting most from the concept?	How appealing is the overall value proposition for the user? What value does this concept offer over the user's existing next-best alternative to drive adoption?	Change resistance What level of resistance might be expected to the change?	User desirability 1 2 3 4 5 1 = Low and 5 = High				
Technical value – Feasibility impact assessment							
What are the implementation and operational risks? From a software engineering, integration, and data perspective?	Are sufficient safeguards available? From an information security, responsible AI, compliance, and intellectual property perspective.	AI/LLM fit Why does this use case require AI/LLM technology as opposed to other alternatives?	Technical feasibility 1 2 3 4 5 1 = Low and 5 = High				

Now let's return to the strategic fit score we calculated earlier, when we identified these use cases. Combining the strategic fit score with the business viability, experience value, and technical feasibility for each of your use cases yields the metrics we use to determine which use case is worth prioritizing. We use two metrics as axis to visualize the viability of these use cases:

1. Degree of strategic business impact = mean of strategic fit and business impact
2. Degree of executional fit = mean of user desirability and technical feasibility

Using these two values, you can graph the results of your use case prioritization:



Where each use case falls can help you determine the best path forward for its development.

- **Shelf:** Low business impact and value, and potential difficulty with development and/or demand, suggesting that this use case is not viable and should be shelved until circumstances change.
- **Research:** Potentially high business impact but currently unfeasible or have low demand. These use cases are ripe for research into factors that may impact the executional fit of this solution such as problem space, user needs, and/or market conditions.
- **Incubate:** Technically feasible and/or have existing user demand, however with low business impact or strategic fit. Use cases falling into this category are ripe for prototyping and testing in controlled environments to help ensure the use case is refined to align with your strategic fit and business goals.
- **Accelerate to MVP:** High strategic business impact, technically feasible, and meet existing demand. These are use cases ready to be invested in and developed.

Example Scenarios: Use Case Prioritization

Besides the two analyzed use cases, another team completed a business envisioning exercise for a third use case: an AI-powered inventory management solution integrated with inventory databases. The table that follows shows its BXT scores. Let's review the results for all use cases, including their initial strategic fit scores and BXT component scores, and visualize their positions on our prioritization chart.

[Expand table](#)

BXT Component	Store Operations Assistant	Shopping Application	Inventory Manager
Strategic fit	4	5	3
Business impact	3	4.3	3
User Experience	3.7	4	3
Technical feasibility	4.3	4.3	4
Degree of strategic business impact	$(4+3)/2 = 3.5$	$(5+4.3)/2 = 4.7$	$(3+3)/2 = 3$
Degree of executional fit	$(3.7+4.3)/2 = 4$	$(4+4.3)/2 = 4.2$	$(3+4)/2 = 3.5$

Using our two equations, we can see where each of these use cases fall on this chart.



We can see that Contoso shopping application rests squarely in the "Accelerate to MVP" quadrant as shown above, suggesting this use case is best suited for development, both technically and strategically. This corner represents the meeting point of the best value and technical feasibility. In contrast, the inventory manager doesn't provide adequate strategic business impact to develop and should be incubated for more testing and revision.

Notably, the Store Operations assistant also falls in this quadrant, suggesting it would also be fit for development. However, the business value and technical potential of the Shopping application are clearly greater. Additionally, the Shopping application provides large learning potential surrounding building larger, more complex

applications. On top of the obvious revenue potential of standalone commercial application, this effort presents long-term strategic and business value through learnings and templated reusable features.

Given the strong viability of the Shopping application, as illustrated through this exercise, Contoso can proceed confidently into the development stage. The next step in this process is to determine what the best approach would be to developing this solution, and what that journey would look like. Over the course of this ISV application development journey, we return to this example scenario, and others, to give you a roadmap to developing your applications from start to finish.

Next Steps

Now that you identified a priority use case, it's time to start thinking about how to design and build your solution. The next step in this path is to review the [Capability Envisioning](#) and ISV Journey page where you can learn about how to decide the best development approach for your selected use case.

Related Links

- [AI Customer Stories | Microsoft AI](#)
- [Creating GenAI Experiences with the Microsoft Cloud: A Guide for ISVs](#)

Feedback

Was this page helpful?



Choosing an approach to AI-enabled application development with Capability Envisioning

Article • 09/20/2024

- [Introduction](#)
 - [Capability Envisioning](#)
 - [Approach considerations](#)
 - [Example scenario](#)
 - [Microsoft development tools](#)
 - [The ISV Journey](#)
-

This article enables Independent Software Vendors (ISVs) to:

- Learn how to align selected AI use cases with an approach to application development
- Understand the ISV Journey and how it can help ISVs develop high-quality applications

Introduction

After using the [business envisioning](#) framework to prioritize your top generative AI use case, the next step is to plan how to build the solution.

On this page, we guide you through choosing the best approach for executing your use case. We outline key considerations, provide resources to help you think through them, and explore the tools Microsoft offers for building your application based on your chosen approach. To help illustrate this process, we walk you through an example scenario, demonstrating how an ISV might evaluate their options and choose the most effective approach. We also provide an overview of the wider development journey, and how it can differ depending on the approach you select.

Capability Envisioning

After selecting a prioritized use case, it's time to decide how to develop it. ISVs can choose from three main approaches: extend or adopt a Microsoft Copilot, build a custom copilot, or create an application on Fabric.

These approaches are not mutually exclusive, but rather offer flexibility through broad spectrum of tools for developing your application. In the following section we explore how each approach fits into the development process, along with tools and considerations for choosing the best fit for your application. Our example scenario demonstrates the range of effort and resources involved in application development, while highlighting how your choice of approach can impact the development process.

Considerations for your approach

To identify the best approach for development, we explore six primary considerations. These considerations help you evaluate the key features of the intended solution and make an informed decision on the approach that best aligns with your goals.

- **Data:** what types and sources of data and information does the application need to engage with?
- **Customization:** how should customers interact with the application and what control do you need over its outputs?
- **Development complexity:** how challenging is it to build the application, and are the necessary resources and expertise available?
- **End-user:** who is the end user and how much technical understanding do they have?
- **Business value:** how does this application provide customers with value and what is the profitability potential?
- **Risk and compliance:** what regulatory requirements and security concerns are relevant for this application?

The following graphic demonstrates how the answers to these questions can help you align with the different approaches to application development. If your use case has a limited scope and minimal need for customizability, extending an existing Microsoft Copilot using low-code tools may be the best approach. Alternatively, as we explore in a moment, a more complex and customizable solution might be better suited to building the application on Fabric.

This use case was prioritized for development using the business, experience, technology framework to evaluate and compare its viability with other potential use cases. Explore how you can prioritize your own use cases in more detail [here](#).

Example scenario: Contoso Shopping application

In our scenario, Contoso partners with retailer AdventureWorks to develop a solution that provides a virtual storefront and systems to improve employee communication, inventory planning, and data insights across operations. Let's review how this use case aligns with our considerations for choosing a development approach.

Considerations

Contoso store operations assistant

- **Data:** The application requires distributed data from a diverse set of sources, including non-Microsoft cloud applications, surfaced via APIs.
- **Customization:** The application is custom built for AdventureWorks with extensive customizability and control over individual features. The application needs to have multiple potentially complex components to support different stakeholders and tasks.
- **Development complexity:** Development of the application's capabilities require substantial resources, time, and human capital, including professional software developers.
- **End-user:** The end-user can vary across AdventureWorks operations but includes technical data scientists and nontechnical frontline workers.
- **Business value:** This application serves as a core feature of AdventureWorks operations by enabling a virtual storefront with substantive revenue potential along with data-driven insights on company operations. For Contoso, this offering represents a major business opportunity.
- **Risk and compliance:** This solution interacts with protected financial data to complete transactions for AdventureWorks customers, necessitating significant security and regulatory compliance components.

Contoso then used these considerations to assess how their development approach aligns with their overall strategy, business value, and technical capabilities, recognizing that alignment in these areas is critical for a successful application. Let's review how each of these considerations impacted their decision.

- **Strategy:** Given the use case prioritized in their business envisioning session, Contoso needed an application with expansive, and modular capabilities that could handle complex and highly customized functions. The Build on Fabric approach is ideal for this scenario because it helps Contoso teams learn new tools and technology, and the application needs features that require extensive development.
- **Business:** In addition to the larger revenue generating potential of this complex application, building the Contoso Shopping application on Fabric means Contoso

can also template components of the application for reuse. With these templates, Contoso can accelerate future development efforts, potentially reducing costs and improving time to value.

- **Technology:** Lastly, and most obviously, the technical parameters of Contoso's use case and AdventureWorks' problems mean the Contoso Shopping application requires substantial pro-code development, along with more advanced data infrastructure and customization. Building an application on Fabric is the best path for Contoso to be sure they have the tools they need to execute on this use case.

In this scenario, Contoso decided to develop the Shopping application using pro-code given the need for extensive customization and the variety of complex features. Low-code development does not enable the capabilities identified in this use case. Within the pro-code journey, Contoso chose to develop this application from the ground up, on Fabric, to ensure a solid foundational data infrastructure on which to build its various features. They were able to make this decision confidently because this approach to development aligned with their use case strategically, commercially, and technically.

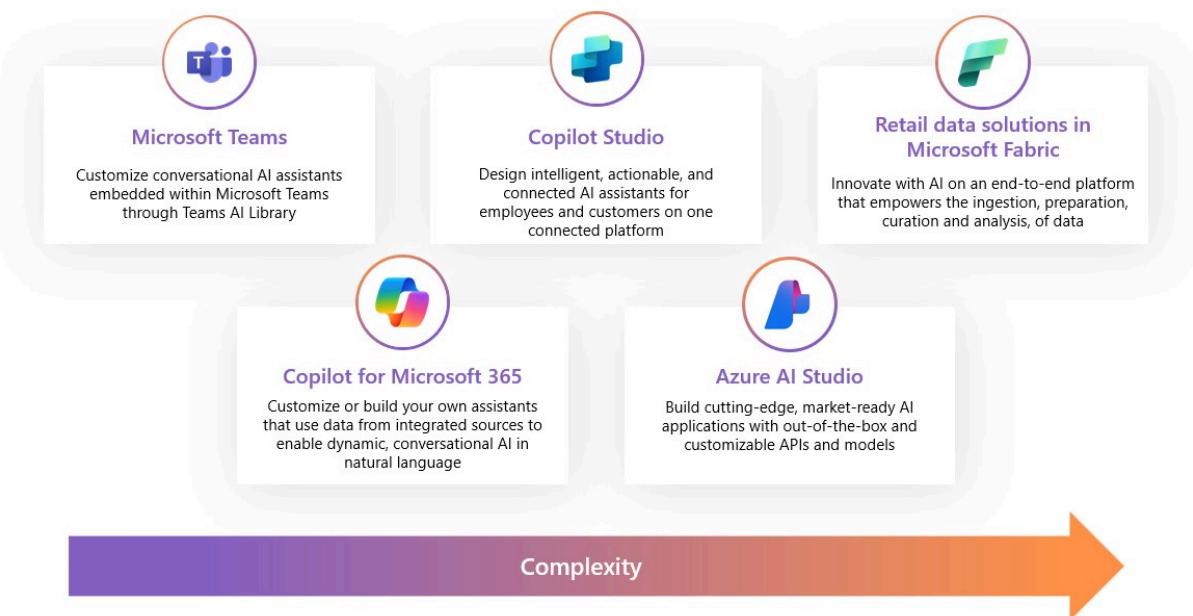
Let's also briefly examine why Contoso didn't choose the adopt/extend a Microsoft Copilot, or build your own copilot approaches. First, adopting and/or extending a Microsoft Copilot limits the application to a conversational assistant, which does not satisfy the requirements of the Contoso use case or meet AdventureWorks' needs. Similarly, while building a custom copilot enables greater customization and complex interactions, the technical data functionality required for a Shopping application extends beyond the capabilities of a copilot on its own. With these considerations, building an application on Fabric is the best choice for this scenario.

Given the application's variety of interfaces and capabilities, Contoso ultimately used multiple tools, including Microsoft Fabric, Microsoft Azure, and Azure AI Studio, to complete development. It's important to note that the three approaches are not mutually exclusive. A custom copilot, built with low-code in Copilot Studio, using the Store Operations copilot template, could easily be integrated into this Shopping application. Selecting an approach is not a walled highway, it is a tool to help think through the requirements of your application, the capabilities you envision for it, and the resources you need to develop it. Let's dig a little bit deeper into how Microsoft solutions fit into these approaches.

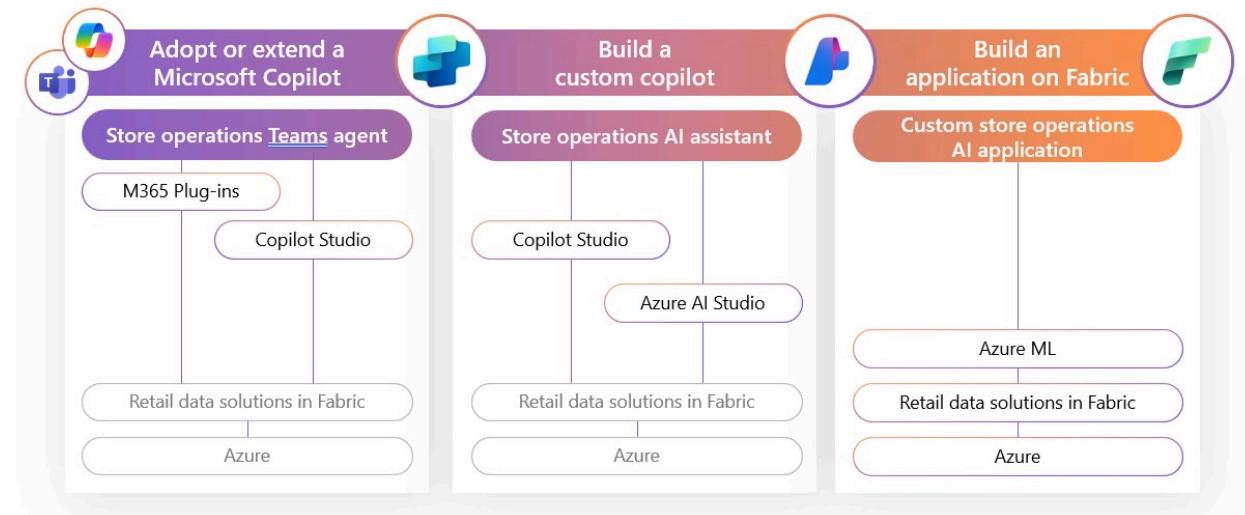
Microsoft Development Tools

Microsoft offers various development tools to meet ISVs' application development needs. Each of these approaches can involve a mixed array of individual Microsoft tools.

We've summarized the main tools. You can also review a [detailed guide for ISVs](#) on building GenAI experiences that explains the possibilities of each approach.



You can see how these solutions map to the approaches in the graphic that follows. We use the store operations application, which we compared to our Shopping application on the business envisioning page as an example use case.



The store operations use case involves an AI assistant that enables rapid access to store procedures, policies, and data in natural language. This use case could be developed using any of the three approaches, to different levels of complexity. An ISV could rapidly develop an assistant to fulfill this use case using the [Copilot Studio Store operations template](#). This effort would require minimal coding and involve interfacing at the surface of the stack, with M365 plug-ins or Copilot Studio, as shown in the left column.

Alternatively, an ISV could undertake a more complex development process and utilize Azure AI studio or Fabric to develop a more comprehensive store operations application, including data infrastructure and technical user assistants. This approach, as in the Shopping application scenario, would involve substantial pro-code development with developers interfacing with more of the full stack, as shown in the right column.

Your specific needs and circumstances determine the best development approach. These tools and methods support you regardless of how you develop your application, but the path varies based on your choice. You can use this  template to walk through the GenAI considerations for your use case and determine the best path forward.

Which generative AI approach should you pursue?

Data	Customizability	Development Complexity	End user	
<ul style="list-style-type: none">• Where is my data?• What infrastructure do I need to connect it?• How do I need my data to interface with Microsoft data and other sources? <p>• [Your responses here]</p>	<ul style="list-style-type: none">• How will customers interact with my app?• Do I need to control its tone and behavior?• Am I prepared to manage security and compliance? <p>• [Your responses here]</p>	<ul style="list-style-type: none">• How much time can I dedicate to app dev?• What resources can I dedicate to this app?• Who do I have available to staff to this project? <p>• [Your responses here]</p>	<ul style="list-style-type: none">• Who is the end user?• How much technical know-how will my end user have?• Where would the end-user interface with the app? <p>• [Your responses here]</p>	<p>Adopt or extend a Copilot</p> <p>Build a custom copilot</p> <p>Build an app on Fabric</p>

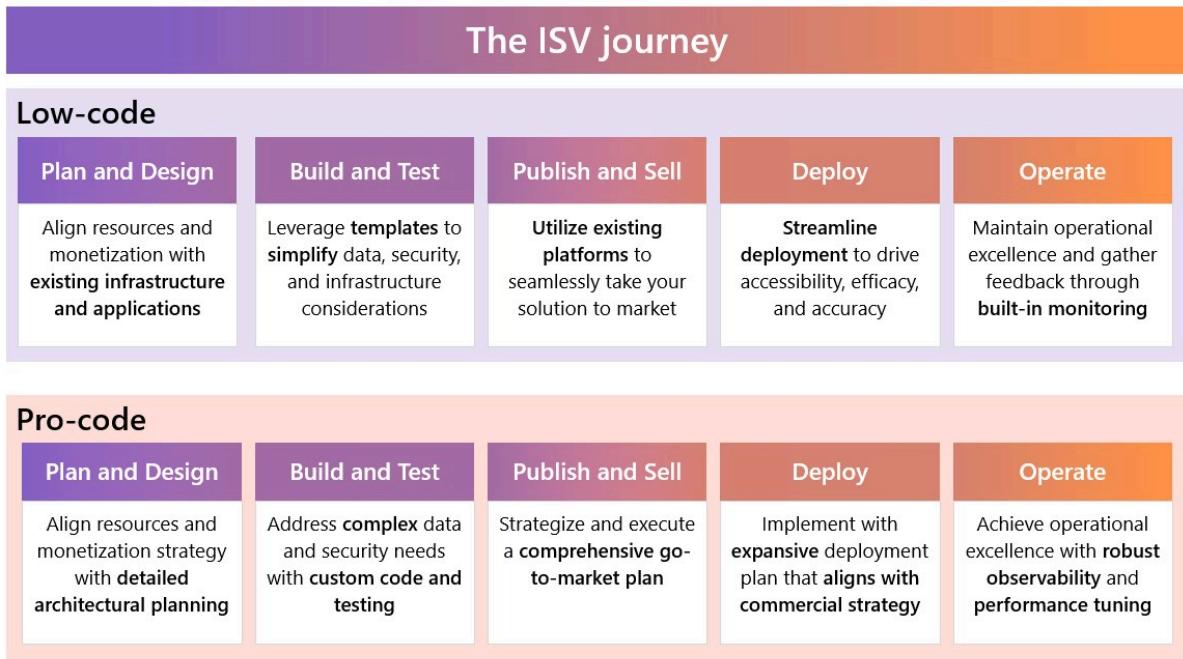
Now that you have identified and evaluated your use case and chosen the best development approach, let's look at the path ahead.

The approaches across the ISV journey

The three approaches to building AI and GenAI applications on Microsoft tools can be split into two primary development journeys: a low-code journey and a pro-code journey. The low-code journey is characterized by rapid time to value and light application development lift while the pro-code journey powers more application customizability and complexity.

Our experience working with ISVs has led Microsoft to develop the ISV Journey Map, a consistent, and systematic process to develop applications for the Microsoft Cloud. This framework is designed to help you reduce costs, and efficiently develop the best possible solution. If you'd like to be evaluated using this framework, contact your partner development manager.

The ISV journey provides a broad overview of the different phases you'll go through to develop your application. Your chosen development approach will shape your experience in each phase, but the overall structure remains consistent across development lifecycles. This applies whether you are extending a Microsoft Copilot with low-code or building an application from scratch with pro-code.



Microsoft continues to build out content to provide holistic guidance for building AI and GenAI applications for the Microsoft Cloud. You can find more content and resources at [this Microsoft Copilot for ISVs Collection](#). This page is regularly updated with newly developed content.

Next steps

Selecting the appropriate approach to your applications development is a critical stage in this process. Whether you need rapid time to value or complex customization, Microsoft is here to help you bring value to your customers. Once you've selected the right approach for your application, then comes the question of choosing the appropriate path for implementation. Each approach includes several potential tools, or patterns, you can use to develop your application. Here are pages where you can find more information about [low-code patterns](#) and [pro-code patterns](#).

Related Content

- [Official Collection | Microsoft Copilot for an Independent Software Vendor \(ISV\)](#)
- [Creating Generative AI Experiences with the Microsoft Cloud: A Guide for ISVs](#)

- Building AI solutions with partners: Empowering transformation with copilots | The Microsoft Cloud Blog ↗
-

Feedback

Was this page helpful?

 Yes

 No

Creating Generative AI Experiences with the Microsoft Cloud: A Guide for ISVs

Article • 09/20/2024

Welcome to your guide to creating unique Generative AI (GenAI) experiences with the Microsoft Cloud. As an **independent software vendor (ISV)**, you're in a prime position to harness the power of GenAI to innovate and deliver captivating solutions to your customers.

What is Microsoft Cloud?

The [Microsoft Cloud](#) is a comprehensive and integrated platform offering a wide range of capabilities and services. It includes Azure AI, Microsoft 365, Microsoft Fabric, and more, putting it at the forefront of the global Generative AI revolution.

This platform allows you to surface your proprietary data and functionality into various areas, including Microsoft 365, a hub of productivity and collaboration accessed by millions.

This guide helps you navigate the expansive possibilities available across the Microsoft Cloud ecosystem.

What are copilots?

We refer to a copilot as an AI-powered virtual assistant that enhances user productivity by assisting humans with complex cognitive tasks, providing contextual suggestions, and driving data-rich insights. These copilots can be grounded in specific customer or ISV data and context, offering an opportunity for ISVs to create generative AI experiences that understand business-specific data.

Scenarios and approaches

Copilot extensibility ISV approaches



Extend Copilot

Surface your data and services into Microsoft's Copilots.



Create plugins
to enhance an existing Copilot's functionality.



Microsoft Graph connectors
Bring in your data to Copilot experiences.



Create copilots

Create copilots anywhere with minimal coding and optional Microsoft data integration.



Power Platform Connectors
Enable customers to create their own copilot experiences with your connectors on Power Platform.



Microsoft Graph APIs
Leverage Microsoft Graph API in your copilot.



Azure OpenAI Assistants
Easily connect your existing APIs with function calling.



Teams AI Library
Use the Teams AI Library to build your own copilot on Microsoft Teams.



Full control

Build your own end-to-end AI experiences.



Azure AI Studio
Build your own copilot.



Semantic Kernel
Build your own copilot.

This guide provides scenario-led guidance to assist ISVs in navigating the expansive field of GenAI on Microsoft Cloud. Our aim is to help you select the most suitable patterns and technologies for your unique requirements, arranged in three high-level approaches to crafting AI experiences.

Our **approaches** are broken into **patterns** based on **scenarios** to help you navigate the most appropriate path for your scenario and requirements.

Important

Please note, these approaches and their patterns **are not mutually exclusive**. They can be combined to create a tailored solution that best fits your unique requirements and scenarios.

Approach 1: Surface your data and services into Microsoft's Copilots:

This approach is designed for ISVs wishing to integrate their data and services into Microsoft's Copilots. The focus is using plugins and Graph connectors to enhance user experiences.

Scenario: I am an ISV where my end users perform work in Microsoft apps such as Teams, Word, Outlook, and they need to...

...access information using natural language interfaces and I have an existing service I want to make available via these Microsoft 365 apps.

Pattern A: Create plugins to augment Copilot's capabilities with ISV data and functionality

...draw insights from our ISV data sources combined with user centric Microsoft Graph data and their organization's line of business data.

Pattern B: Use Graph connectors to make ISV data available for Copilot experiences.

Approach 2: Create copilots anywhere, with minimal coding and optional Microsoft data integration:

This approach is for ISVs aiming to enrich their apps with Microsoft's data and tools, or who want to create their own AI assistants with Azure. It involves making use of the Microsoft Graph API, Copilot Studio plugins, Teams AI Library, or enabling customers to create their own copilot experiences with your data via connectors.

Scenario: I'm an ISV where my end users work anywhere...

...and I want to enable them to create their own copilot experiences using our data and services.

Pattern C: Develop Power Platform Connectors to enable customer-driven copilots in Copilot Studio

Scenario: I'm an ISV where my end users work in my existing applications and UI, who want us to...

...incorporate Microsoft user-centric **Graph** data into my copilot.

Pattern D: Use Microsoft Graph API in your copilots

...provide a conversation experience within my existing application, which can answer questions and turn conversations into actions.

Pattern E: Create your own AI Assistants with Azure

Scenario: I'm an ISV who's Copilot experience targets **Microsoft Teams** and includes requirements to...

...create smart Teams bots that integrate to GenAI, run bots in Teams and have context-aware conversations or use Teams chat experience and I as an ISV focus on complex business logic.

Pattern F: Use the Teams AI Library to Build Your Own copilot

Approach 3: Full control: Build Your Own (BYO) end-to-end AI experiences:

This approach is geared towards ISVs seeking to craft entirely new AI experiences or copilots using tools such as Azure AI Studio and Semantic Kernel. It offers maximum control and customization, providing limitless possibilities.

Scenario: I'm an ISV seeking to develop cutting-edge interoperable AI experiences that...

...require specialized handling of intricate, multimodal data, which might include fine-tuning AI models to meet specific accuracy needs.

...use ISV's custom AI models and services for tailored solutions.

...provide complete control over the conversational experience, including system prompts, temperature, tone, and custom safety requirements.

Pattern G: Build new interoperable AI experiences or copilots using your existing APIs using Azure AI Studio

Pattern H: Build your own copilot with Semantic Kernel

Tip

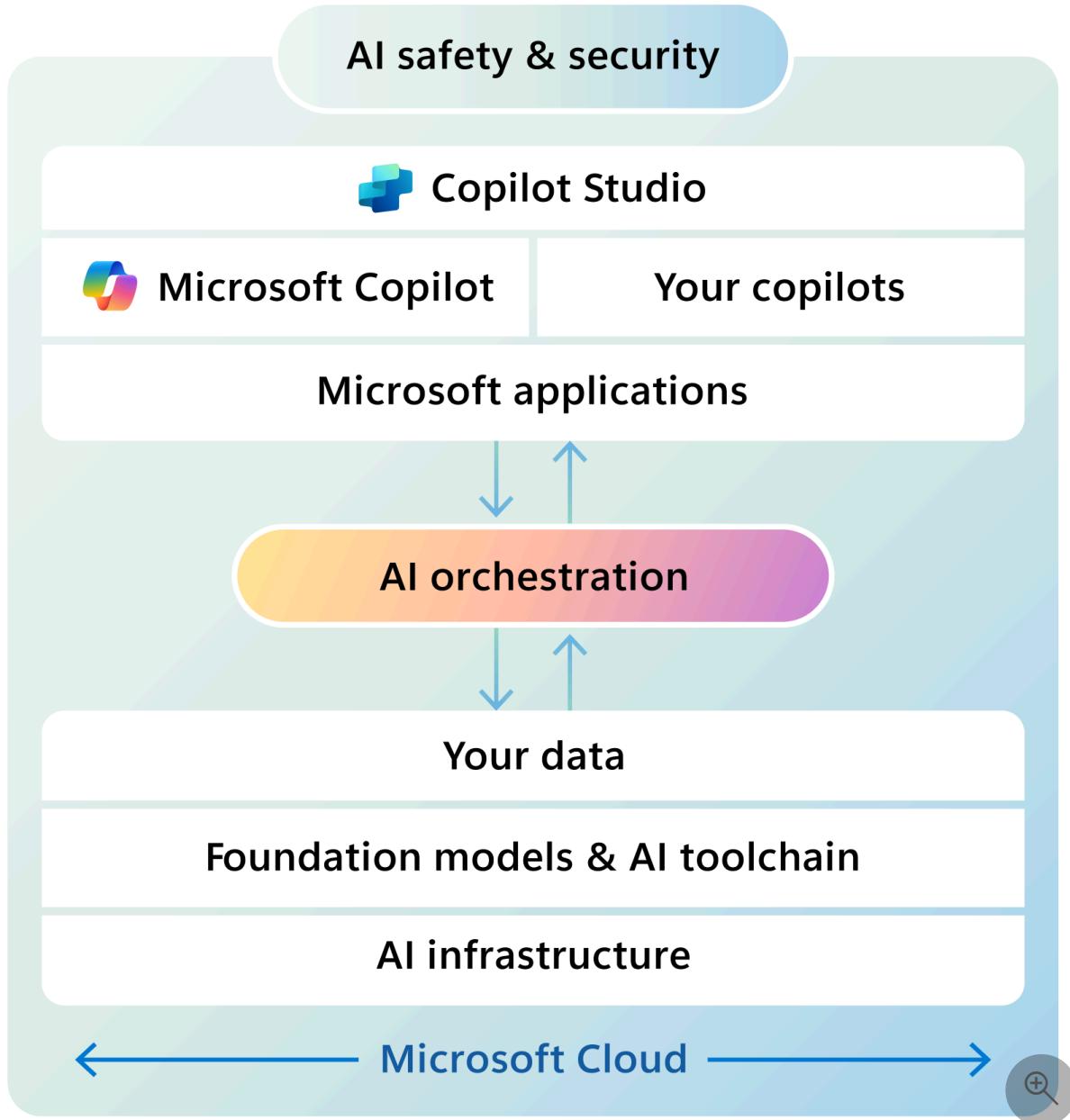
For a deeper understanding, we encourage you to explore the detailed content available in the **Get Started** links provided in each pattern.

The Microsoft Copilot stack comprises three distinct tiers: the back end (with the data sitting in respective repositories), an AI orchestration tier in the middle, and the front end (UI experience of your copilot). Within each tier, there are recommended layers that should be considered when building a copilot.

As ISVs move from Approach 1 to Approach 3, they engage more deeply with each tier, assuming greater development responsibilities. For example, creating a plugin under Approach 1 means Microsoft handles AI orchestration, including data integration and

Responsible AI. Conversely, developing a custom copilot might require full control over the user experience, orchestration layer, data management, and Responsible AI.

Copilot stack



ⓘ Note

While "copilot" refers to the general concept of a generative AI assistant, "Copilot" refers to specific Microsoft products, such as Microsoft 365 Copilot or Dynamics 365 Copilot that ISVs can integrate with.

Each option varies in complexity and effort. Adopting existing Microsoft Copilots is straightforward, extending them with plugins requires minimal effort, and crafting a new copilot experience might need design, science, and engineering.

It's important to remember that an AI solution is as good as the data you ground your models on and use as the context. Ready-to-use Microsoft Copilots already support a range of scenarios and can be extended with your data, functions, and processes. However, the user interface can't be extended. Therefore, it's important to carefully consider your specific scenario, how you can apply GenAI algorithms, and how your user (the "pilot") can benefit from your copilot capability.

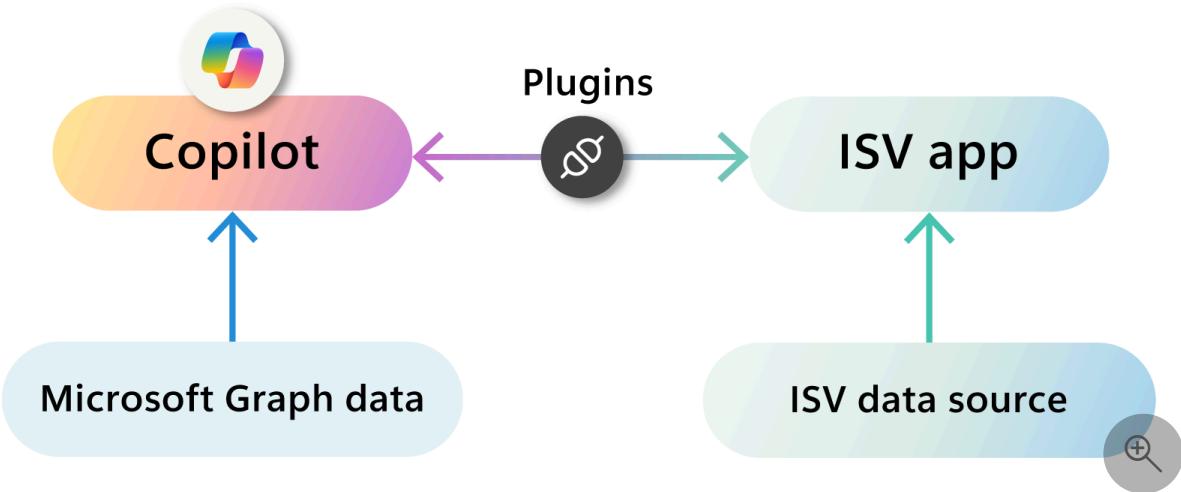
Approach 1: Surface your data and services into Microsoft's Copilots

ISVs looking to surface their existing services, data, and processes into Microsoft's Copilots or Microsoft 365 applications can do so by building plugins and connectors.

This approach allows for example, Microsoft 365 Copilot to interact with APIs from other software and services, surfacing up-to-date information, execute actions, and performing new types of computations.

Pattern A: Create plugins to enhance an existing Copilot's functionality

Plugins are extensions that augment the capabilities of an existing Copilot, allowing it to interact with ISV apps and services. They can expand a user's capabilities by enabling the interaction with your APIs, via natural language conversation. For example, a plugin could allow Copilot to retrieve useful information, perform new computations, or safely execute actions on the user's behalf.



ISVs can create plugins using various tools, including Teams Message Extensions and Power Platform plugins through Copilot Studio. New plugins can be published to Microsoft's Copilot ecosystem via Partner Center, where IT admins can approve them for use by end users.

ISV scenarios:

- ISVs looking to surface their existing services on Microsoft 365 client apps
- Users can search, update, and perform actions in an ISV application or any external systems, from Microsoft 365 client apps, such as Teams, Outlook, Word etc.
- A user on Teams could find relevant information from an external ticketing or CRM system your customers use, optionally including executing actions on the user's behalf, within the system

Partner advantage:

- Service the millions of users and companies using Microsoft 365 client apps, meet users where they work right now.
- Increase your solution service visibility by surfacing them in Microsoft 365 client apps.
- Reduce your users' friction by eliminating the need to navigate between multiple apps and canvases.
- A consistent user experience across Microsoft 365 apps with universal integration and continuity across Microsoft 365 apps.
- For example, a Contoso user discovers latest customer account summarized with data coming from Outlook, SharePoint and Fabrikam's external system, without leaving Teams.

Where to start:

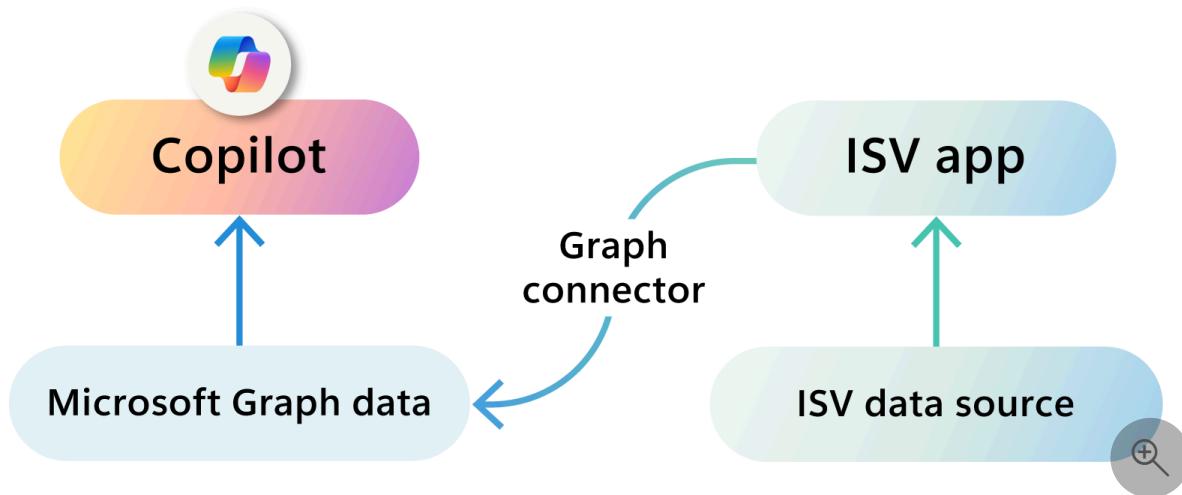
- Build message extensions for Microsoft Copilot for Microsoft 365 | Microsoft Learn
- Create copilot plugins - overview (preview) - Microsoft Copilot Studio | Microsoft Learn

Key takeaways:

- Plugins are a way to surface ISV services and apps on Microsoft Copilots allowing end users interact with ISV apps and services from Microsoft 365 client apps.
- ISVs can create plugins using tools like Teams Message Extensions and Copilot Studio plugins.
- Plugins can increase your solution's visibility and discoverability through Partner Center

Pattern B: Use Graph Connectors to bring your data to Copilot experiences

[Graph connectors](#) enable ISVs to connect their data to the Microsoft 365 Semantic Index. Their data becomes searchable and actionable for users, directly from Microsoft 365 client apps such as Teams, Outlook, and Word. Microsoft Copilot becomes grounded in ISV data, whether cloud or on-premises, via the Microsoft Graph. Furthermore, ISVs can use [Microsoft Fabric](#), a unified data platform delivered as a SaaS product, to bring their data into the Microsoft Cloud and easily connect it to the Microsoft Graph.



ISVs can build Graph connectors using the Microsoft Graph Connectors API, which supports a range of data sources, file systems, web pages, enterprise applications, and more.

Graph connectors can also enrich the data with AI-powered capabilities, such as natural language processing, entity extraction, and image analysis. By using Graph connectors, ISVs can extend Microsoft Copilot with their own data, enhancing the user experience and enabling more personalized and secure interactions.

[Hundreds of Graph connectors already exist](#). For example, the Jira Cloud graph connector can elevate Jira objects to the same level as Microsoft 365 Graph data, allowing comprehensive reasoning and universal integration, leading to enhanced and richer insights. The connector allows end users to search for Jira objects from Microsoft 365 Copilot using natural language interface.

ISV scenarios:

- ISVs whose customers perform work in Microsoft 365 and want to enable end-users to draw insights from ISV data sources combined with user centric Microsoft Graph data.
- Users can retrieve, summarize, and reason over data from ISV applications, combined with other Microsoft 365 graph data, for example, emails, word documents etc.
- Communications Director needs to find and redraft PR emails in Outlook that are enriched with content sourced from an ISV graphics design application

Partner advantage:

- Meet users where they work. A vast user base uses Microsoft 365 client applications and now can access Your own data and service in one unified experience.
- Enriched insights by combining ISV data with Microsoft Graph data.
- Universal integration with Microsoft 365 apps, and Microsoft Search, Context IQ and Viva, with one connector.
- Expanded channels to showcase ISV data, potentially increasing user base.

Where to start:

- Learn more about graph connectors at aka.ms/graph-connectors
- [Build your own graph connector at Build Microsoft Graph connectors for Microsoft Copilot for Microsoft 365 | Microsoft Learn](#)

Key takeaways:

- Graph connectors allow ISVs to bring their data into the Microsoft Copilot ecosystem, enhancing the user experience with personalized and secure interactions.

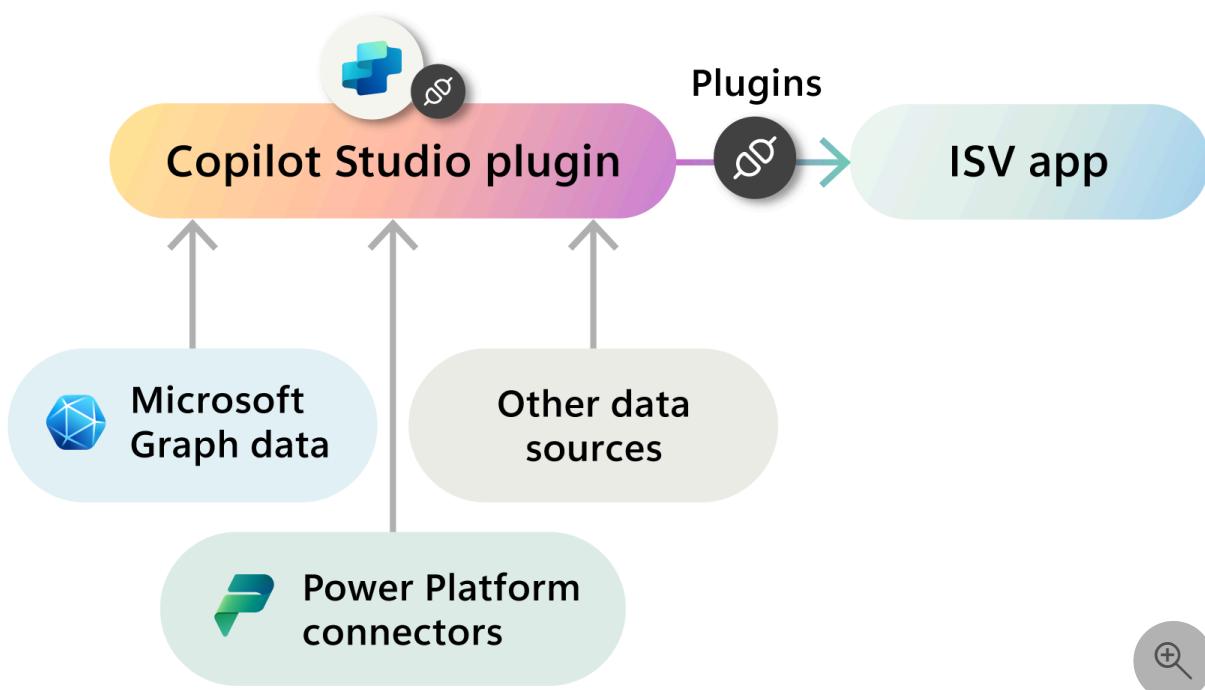
- ISVs can use Microsoft Fabric to bring their data into the Microsoft Cloud and connect it to the Microsoft Graph ↗.
- By using Graph connectors, ISVs can combine their data with Microsoft 365 graph data to provide enriched insights and achieve universal integration with Microsoft 365 apps.

Approach 2: Create copilots anywhere with minimal coding and optional Microsoft data integration

ISVs can bring the power of tools and data sitting in Microsoft Graph into their own apps, enhancing their functionality and user experience.

Pattern C: Develop Power Platform Connectors to enable customer-driven copilots in Copilot Studio

Copilot Studio enables customers to create low-code AI apps that can respond to common user queries, using data from their organization and Microsoft and partner data sources. Copilot Studio uses Power Platform Connectors to bring in data from potentially any source, where there are more than 500 connectors today. As an ISV you can create connectors to your data and services, to empower your customers to create their own internal copilots and AI apps, grounded in ISV data.



These AI apps can be surfaced to end users across various platforms, including websites, mobile apps, Microsoft Teams, or any channel supported by the Azure Bot Framework.

Example scenarios:

- ISVs looking to provide a chat bot experience to their customers, within their existing applications, which can respond to questions and turn conversations into actions.
- Users asking questions within your application and receiving answers grounded in ISV, Microsoft, or customer data sources.
- Create customer connector from your existing APIs and convert it to plugin using Copilot Studio allowing these APIs to be called from a chat bot with natural language interface.
- Convert existing Power Automate flows into plugins that can be called from Microsoft copilot chat to perform actions and retrieve information.
- Access data through natural language interface from enterprise systems such as Zendesk, GitHub, and Salesforce through these connectors in Power Platform.

Partner advantage:

- Harness the power of existing Microsoft and non-Microsoft connectors to enhance and enrich your application effortlessly.
- Expedite plugin development by applying Power Platform custom connector approach for swift and efficient integration.
- Enhance time-to-value through the low-code capabilities of Copilot Studio.
- Gain a competitive edge by integrating AI capabilities into your app with minimal coding.

Where to start:

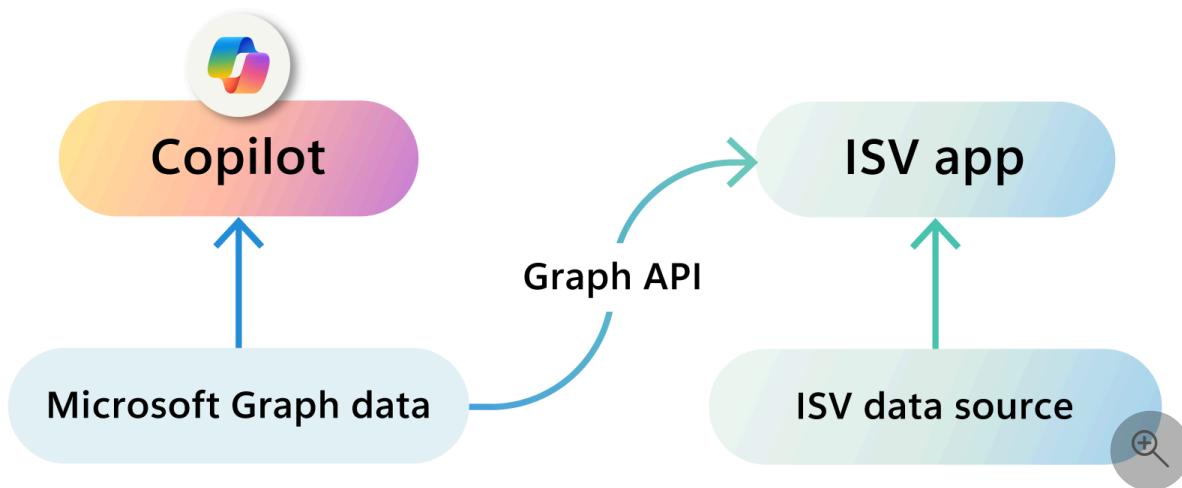
- [Microsoft Copilot Studio plugin architecture - Microsoft Copilot Studio | Microsoft Learn](#)
- [Embed a Power Virtual Agents control using chatbot control | Microsoft Learn](#)

Key takeaways:

- Copilot Studio offers a platform for creating low-code AI apps that can enhance existing applications with chatbot capabilities grounded in ISV data or functions.
- The platform supports both existing Power Platform Connectors and custom connectors, offering flexibility in integrating ISV services and data sources.
- The integration of AI capabilities can significantly improve the user experience and give your app a competitive edge.

Pattern D: Leverage Microsoft Graph API in your copilots

The Microsoft Graph API offers a powerful endpoint to access user-centric data from Microsoft 365 applications, which includes Calendar, Bookings, Outlook, Teams, OneDrive, SharePoint, and [more](#). With this API, you can enrich your apps with data from Microsoft 365, enabling users to derive richer insights and analytics.



ISV scenarios:

- Customer and partners using existing ISV application looking to combine Microsoft user centric Graph data in their copilot.
- An ISV with a project management app wants to incorporate Microsoft 365 calendar data and project documents data to help users track deadlines and milestones within the app.
- An ISV with a CRM app wants to incorporate Microsoft 365 contact and email data to enhance customer profiles and communication logs.

Consider Fabrikam, a versatile Human Capital Management (HCM) software equipped with a flexible HR suite, empowering seamless automation of various workflows such as talent acquisition, employee rewards management, and feedback processes. In their continuous pursuit of innovation, Fabrikam introduces a cutting-edge copilot feature atop their HR suite. Now, they aim to elevate their application even further by integrating user-centric graph data. This enhancement involves using Graph API to incorporate employees' calendars, encompassing details like scheduled time-offs and 1:1s for feedback processes etc.

Partner advantage:

- Uncover enriched insights by combining your data with Microsoft 365 Graph.

- Seamless Integration: Standardized access to Microsoft 365 data for easier integration with your apps.
- Improved User Experience: Provide a more seamless user experience with access to relevant Microsoft 365 data and features within your app.
- Enhanced Functionality: Add new features and capabilities to your app using Microsoft 365 data.
- Scalability and Efficiency: Focus on building and improving your apps while the Graph API handles data retrieval.

Where to start:

- [Use Graph Explorer to try Microsoft Graph APIs - Microsoft Graph | Microsoft Learn](#)
- [Quick Start - Microsoft Graph](#)

Key takeaways:

- The Microsoft Graph API allows ISVs to enrich their apps with user-centric data from Microsoft 365.
- Via Graph APIs you can leverage the Microsoft 365 Semantic Index, a more advanced search experience built for the era of Copilots.
- By using the Graph API, ISVs can enhance their apps with richer insights and analytics.

Pattern E: Bring a copilot experience to your apps with Azure OpenAI Assistants

ISVs can adopt this low-code approach in Azure's AI Services to bring copilot-like experiences to their own applications. It offers a fast path to apply GPT's [function calling](#) to call your own APIs simply by describing your function's structure in JSON and providing a sandboxed python environment to [run and execute code](#) to help formulating responses to user's questions.

Both these features can be useful in offloading non-language-based challenges to conventional code or existing systems that are better suited for the task, for example simple maths tasks.

While you don't have direct access to the system prompt and temperature, you can similarly affect the behavior of your Assistant via Custom Instructions that have a heavy influence on the personality of your copilot-like experience.

Partner advantage:

- Azure OpenAI Assistants provide a low-code approach, enabling ISVs to quickly integrate Generative AI capabilities into their applications without extensive development effort.

Where to start:

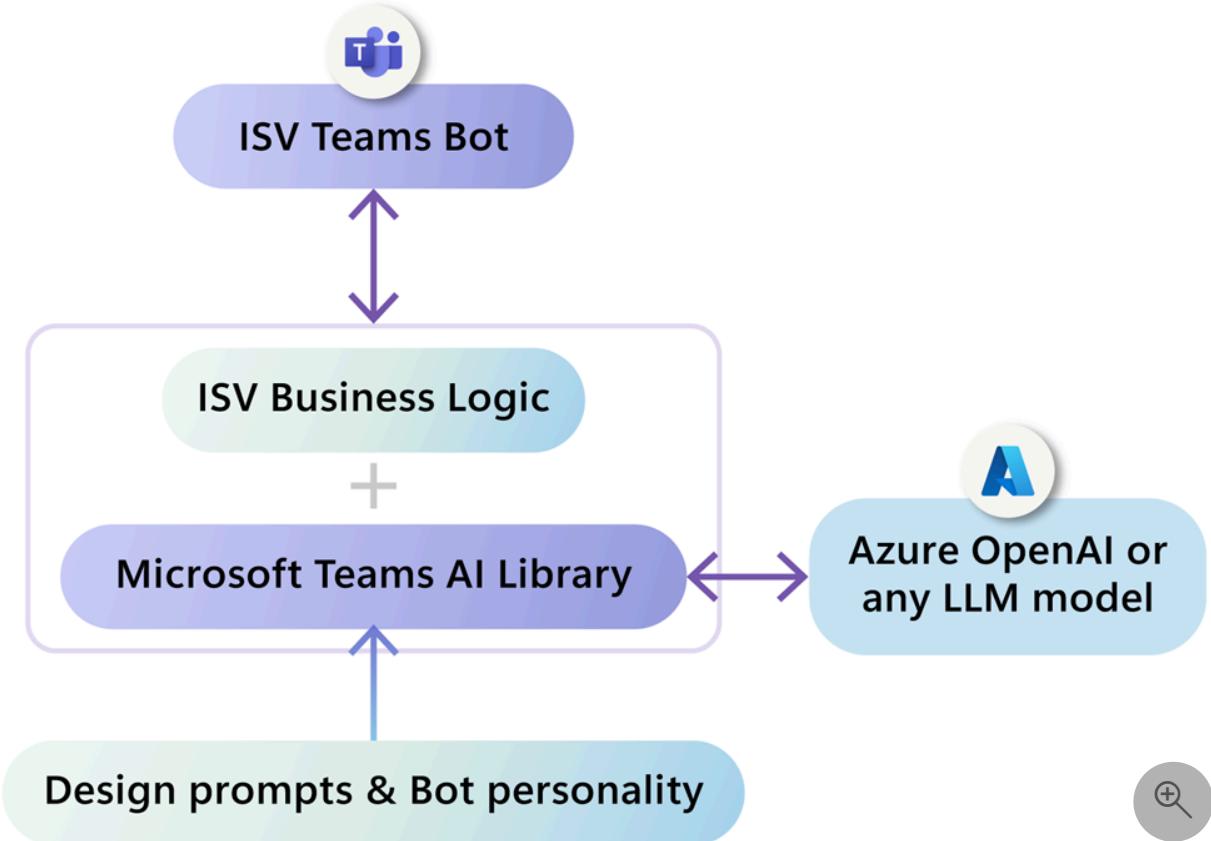
- [Quickstart - Getting started with Azure OpenAI Assistants \(Preview\) - Azure OpenAI | Microsoft Learn](#)
- [How to create Assistants with Azure OpenAI Service - Azure OpenAI | Microsoft Learn](#)

Key takeaways:

- ISVs can use Azure OpenAI Assistants to create interactive, natural language interfaces that enhance user engagement. These assistants can call out to APIs via simply by describing them via JSON.
- An Azure OpenAI Assistant can write and execute code, in a sandbox, based on a user's prompt, to solve a non GenAI problem.

Pattern F: Use the Teams AI Library to build your own copilot

ISVs can also use the [Teams AI Library](#) to add natural language capability in their existing Teams chatbot. This library allows ISVs to focus on their business logic, while using the Teams scaffolding to handle conversational interactions. ISVs can surface their chat bots in Teams, offering users a more natural and intuitive way to interact with their apps.



ISV scenario:

- End users are using Teams and ISV partner is looking to surface their service or functionality on Teams with Bot-like capabilities.
- No integration is needed with Graph data and ISV partner is looking to focus on the service and business logic without integrating with Teams Copilot capabilities.
- With prebuilt Teams app templates and built in moderation safety features, ISV partner can easily add LLM capability to their existing chat bot.

Partner advantage:

- Add ChatGPT like conversational experiences, with control over prompt engineering to your bot and reuse built-in safety features.
- Built on top ready to reuse capabilities like
 - Conversation session history offered by Teams AI mechanism.
 - Multi-language support.
 - Multi Large Language Models support, beyond OpenAI models.
 - Action planner that can help map to actions based on the user intent.
 - Ready to use augmentation mechanism to change the way model is responding through parameters or system prompt change.
 - Extra reasoning that can ground the answers from the model on your data.

Where to start:

- [Introduction to Teams AI Library: Teams AI library - Teams | Microsoft Learn](#)
- [List of technical capabilities: Teams AI library capabilities - Teams | Microsoft Learn](#)

Key takeaways:

- Team AI library provides an easy way to light up an ISV developed bot in Teams with the power of LLMs.
- It doesn't require the integration with current Microsoft Copilot capabilities, can provide a task-oriented experience.
- It offers many possibilities from an engineering perspective but also ready to use capabilities Out Of the Box, making the whole development process easier.

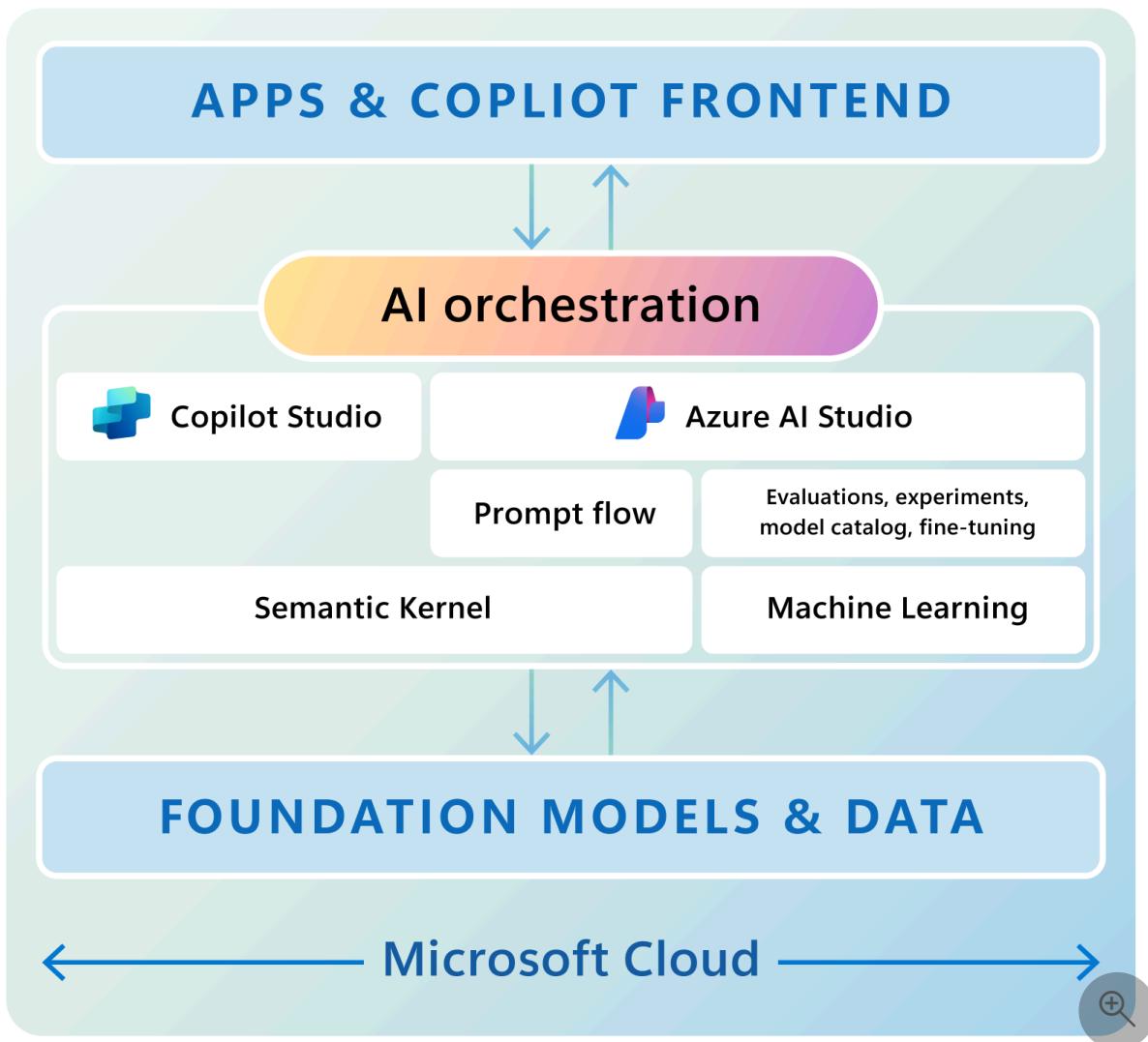
If you want to power your bot in Teams with LLMs, Teams AI Library is the way for you to go.

Approach 3: Full control: Build Your Own (BYO) end-to-end AI experiences

ISVs can use the Microsoft Copilot Stack to build entirely new AI experiences, as copilots or intelligent assistants. An ISV building in this middle part of the stack takes responsibility for AI Orchestration - where Microsoft offer various options, all of which apply Microsoft's foundational models, AI toolchain and AI infrastructure.

[Semantic Kernel](#) can be leveraged to build the same AI orchestration patterns that powers Microsoft Copilots, in your copilots. It's available as [an SDK you can develop against directly](#).

Copilot stack orchestration



With most the investment creating a copilot service in the middle of the stack, ISVs have the freedom to connect this copilot service to various surfaces, including Teams, Microsoft 365 Copilot, Microsoft Copilot, your own application surfaces, websites, chat bots – or all. Essentially, when it comes to integration with an application surface – the top of our stack – every other pattern described here is also an option.

Pattern G: Build your own copilot with Azure AI Studio

[Azure AI Studio](#) is an all-in-one platform for ISVs to build custom, intelligent assistants, or copilots. It combines capabilities from various Azure AI services, providing a unified workspace for developing and deploying generative AI applications. It's a collaborative platform where data scientists, developers, and other stakeholders can converge and work together.

With Azure AI Studio, ISVs gain full control over their copilot's behavior, personality, and capabilities. Whether using existing pretrained models from our [extensive catalog](#), fine-tuning models on your data, or training your own custom AI models, Azure AI Studio accelerates the development of AI experiences that handle complex multimodal data.

A standout feature of Azure AI Studio is its diverse range of models, catering to various industries and use cases. It allows ISVs to combine different models within a single solution to meet their unique requirements.

Integration with [Azure AI Search](#) enables ISVs to implement a Retrieval Augmented Generation (RAG) pattern for unstructured data directly from Azure AI Studio, with the added advantage of AI Search's Integrated Vectorization feature. This means any data your copilot needs can be automatically kept up-to-date in a vector database, facilitating fast and efficient retrieval during user prompt evaluation, saving you the task of implementing an indexing, chunking, embedding, and vectorizing pattern yourself.

[Prompt Flow](#), a feature of Azure AI Studio, offers a visualized graph for orchestrating executable flows with Large Language Models (LLMs), prompts, and Python tools. It facilitates debugging, sharing, and iterating on your flows with ease through team collaboration.

For ISV teams who prefer a code-first approach, the [Azure AI SDK](#) offers a suite of packages for accessing Azure AI services, including the setup of Azure AI Studio projects and related resources. This allows developers and data scientists to manage AI components, configure AI models, pipelines, and services directly from code, while still making the graphical interface available for those who prefer it.

Prototyping is easy in Azure AI Studio via its Playground. A typical journey for a team working on a project in Azure AI Studio could start with an individual validating an idea in the Playground. Once attractive results are produced, they can be prompted from the Playground to Prompt Flow as a versioned and customized flow. Now a versioned artifact in the AI Project, the wider team can contribute where the flow is accessible via Azure AI Studio UI and via code-only. Multiple branches of logic to differing LLMs can be tested and evaluated at this point.

Beyond the development phase, Azure AI Studio also provides an LLMOps toolchain, handling your end-to-end prompt engineering from development to production and ongoing maintenance.

Azure AI Studio supports integration with Azure AI Search, Azure Open AI Service, and other Azure AI services, simplifying resource management for ISVs. It also provides a project-oriented workspace, fostering collaboration against shared compute, model deployments, and services.

ISV scenarios:

- A healthcare ISV building a telemedicine platform wants a copilot that understands medical jargon, assists doctors in diagnosing patients, and provides relevant treatment recommendations.
- A financial services provider needs a copilot that can analyze market trends, answer customer queries about investment options, and generate personalized financial reports.
- An e-learning platform wants a copilot that tutors' students, explains complex concepts, and adapts its teaching style based on individual learning preferences.
- An insurance company speeds up documents analysis during the claim process by validating if current claim can be covered by the contract.
- Airline copilot can help you plan the journey, look for the tickets and hotels and book them once you are satisfied with the offer.
- A chain of restaurants is creating a copilot app to help new employees to get onboarded by guiding them through the whole process.
- An ISV offers their customers a VS Code extension to help developers build the integration with their APIs.

Partner advantage:

- Customization and Control: Build a bespoke copilot that aligns precisely with your application's requirements.
- Scenario Flexibility: Cater to a wide range of scenarios, from domain-specific copilots to task automation and content generation.
- Integration with Existing Systems: Connect to databases, APIs, and other services to enhance your copilot's capabilities.
- Brand Identity and User Experience: Shape your copilot's personality to align with your brand voice and enhance the user experience.
- Build experience: Open-source and highly extensible SDK, Semantic Kernel let you build intelligent agents that can call your existing APIs. With Semantic Kernel, you can use the same AI orchestration patterns that power Microsoft' copilots in our own apps.
- Scalability and Deployment: Deploy your copilot across multiple clients or applications, serving thousands of users simultaneously.

Where to start:

- [What is AI Studio? - Azure AI Studio | Microsoft Learn](#)
- [Build and deploy your own copilot with Prompt Flow in Azure AI Studio | Microsoft Learn](#)
- [Build and deploy your own copilot with the Azure AI CLI and SDK | Microsoft Learn](#)

Key takeaways:

- Azure AI Studio offers a powerful platform for creating custom, intelligent assistants, or copilots.
- ISVs can shape their copilot's behavior, personality, and capabilities, creating a truly bespoke solution.
- Azure AI Studio supports a wide range of scenarios and integrates seamlessly with existing infrastructure.
- Creating a custom copilot with Azure AI Studio can enhance the user experience and provide tailored solutions for specific use cases.
- AI Studio delivers you a copilot service (or backend), surfaced as a single scaled endpoint
- An ISV then has options to connect the service to an app, front end, or conversation surface of their choice, including any of the previous patterns above.
- Remember, this pattern can be combined with others based on your specific needs. For instance, you might want to pair this pattern with Pattern A and plug-in to a Microsoft Copilot, or Pattern F to surface your own copilot bot in Teams.

Pattern H: Build your own copilot with Semantic Kernel

Semantic Kernel is an open-source SDK that empowers developers to create sophisticated copilots within their applications. It supports a range of programming languages, including C#, Java, and Python, making it accessible to a wide developer community. Semantic Kernel enables the orchestration of AI plugins, allowing for integration with various AI models, including from Azure OpenAI and Hugging Face.

Semantic Kernel encapsulates the essence of Microsoft Copilots' AI orchestration patterns, providing developers with tools to build **agents** and **copilots**.

Agents are AI systems that can answer questions and automate processes for users. They range from simple chatbots to fully automated AI assistants. Copilots, a special type of agent, work alongside users. Unlike fully automated agents, copilots provide suggestions and recommendations, allowing users to retain control.

Plugins: These provide skills to your agent. You can create plugins for tasks like sending emails, retrieving information from databases, or asking for help.

Planners: Agents use planners to generate plans for completing tasks. For instance, a copilot helping a user write an email would create a plan with steps like gathering recipient details and composing the email.

The SDK comes with VS Code extension, sample [Chat Copilot](#) app but also with starters to offer you a scaffolding to bring your ideas to live.

Once you decided to start working with Semantic Kernel, we suggest defining couple of capabilities before starting to code:

- Start by defining a copilot's persona and behavior.
- Create plugins for common tasks your copilot will assist with.
- Use planners to generate plans for copilot actions.
- Plan to test thoroughly to ensure a refined user experience.
- Make sure you're able to gather feedback from your users and implement this in the behavior of the agent or copilot.

ISV scenarios:

- You're building a copilot that is a part of your own application (customer development tool or HR system), and you want people to stay in the realm of the same UI.
- You need a full control on the orchestration engine, RAG implementation, model choices and model parameters.
- With Your copilot service, you want to allow your customers to build extension on top of your solution through plugins.
- Your solution utilizes canvases and other media than just text.

Partner advantage:

- Full control over your copilot behavior with access to opinionated orchestration engine used by Microsoft to build first party Copilots.
- Seamlessly ground models on your own enterprise data and integrate structured, unstructured, and real-time data using Microsoft Fabric OneLake. This allows developers to employ sophisticated hybrid and semantic search to power retrieval augmented generation (RAG) applications.
- Access to superior tools for refining AI responses using prompt engineering and LLM Ops tools like prompt flow.

Where to start:

- [GitHub - microsoft/semantic-kernel: Integrate cutting-edge LLM technology quickly and easily into your apps ↗](#)
- [Building agents and copilots with Semantic Kernel | Microsoft Learn](#)
- [Understanding AI plugins in Semantic Kernel and beyond | Microsoft Learn](#)

Key takeaways:

- Semantic Kernel is an opinionated open-source framework helping developers to build GenAI capabilities into their apps easier

- It's being maintained and developed by Microsoft and used by first party teams to build Microsoft Copilot solutions.
- With set of samples, it helps you to start easily your GenAI journey inside your own application stack.
- Remember, this pattern can be combined with others based on your specific needs. For instance, you might want to pair this pattern with Pattern A and plug-in to a Microsoft Copilot, or Pattern F to surface your own copilot bot in Teams.

Conclusion

We've started with scenarios and brought you to one or more patterns of interest, which we've collected into one of three Approaches. While each pattern has some variance, there are some common features for each Approach:

[\[+\] Expand table](#)

	Approach 1: Surface your data and services into Microsoft's Copilots	Approach 2: Create copilots anywhere with minimal coding and optional Microsoft data integration	Approach 3: Full control: Build Your Own (BYO) end-to-end AI experiences
Development effort	Low (No/low code)	Medium (Minimal code)	High (Pro code)
Data sources	Microsoft Graph (Microsoft/M365 or non-Microsoft via connectors)	Various. Power Platform connectors, Microsoft Graph, Your APIs.	Can span multiple data sources, service, and apps inside or outside of Microsoft tenant
User Interface or conversational surface	Provided by Copilot being extended, for example, Teams, Microsoft 365 etc.	Varies per approach from Provided by Microsoft, to bring your own.	Bring your own. Multiple surfaces possible with same copilot
Influence over copilot's tone, behavior, and model parameters	No direct control. Model parameters are responsibility of Copilot being extended.	Some influence especially for behavior and tone, via custom instructions that form part of the metaprompt. With Teams AI Library, you can control model parameters.	Direct control of model parameters such as temperature, system prompts, max tokens etc. Custom copilot behavior.
Multi-model capable	No	No	Yes Multiple calls to diverse models within

	Approach 1: Surface your data and services into Microsoft's Copilots	Approach 2: Create copilots anywhere with minimal coding and optional Microsoft data integration	Approach 3: Full control: Build Your Own (BYO) end-to-end AI experiences
	same flow		
Model support	Provided by system	Choice of OpenAI models	Choice of any model OpenAI and full model catalog
Responsible AI	Provided by system	Either provided by system or leverageable options in each pattern	ISV responsibility with platform options in each pattern.
Support for chat history	Provided by system	Either provided by system or leverageable options in each pattern	ISV responsibility with platform options in each pattern.
Example scenarios	Users in Microsoft Copilots can perform actions on, or gain insights from ISV data and services.	Introduce a GenAI assistant in existing ISV application surface to reason over customer or ISV data. Present your own copilot or chatbot in an existing Microsoft surface, such as Teams, with a separate identity and experience to Microsoft Copilots.	Your customers and users interact with a fully customized copilot to your brand and behavior , which can reason over multiple data sets and connected systems from a multiple choice of UI or conversational surfaces.

These approaches are in order of increasing possibilities to customize, which also requires an ISV to pick up more responsibility via the control gained and increases the overall development effort.

We therefore highly recommend starting from Approach 1, which might well be the fastest way to market for your initial requirements. Microsoft is releasing new first party Copilots often. Continually check in to see if a new Copilot might address your users needs more efficiently by extending your data and services to it.

Move to Approaches 2 and then 3 gradually, as your requirements lead you to the need for more control and customization.

An exception here is perhaps where an ISV already has an existing AI capability in house with existing assets. For example, an ISV who already has a GenAI team with existing AIOps processes and already has IP created in say Python or LangChain, might be naturally better orientated to Approach 3.

A final key callout is that this list of patterns isn't exhaustive or mutually exclusive. We have curated here select patterns where we see synergies for ISVs and it's important to understand they can be combined in various ways to create a solution that perfectly fits your needs. For instance, while working with Approach 3 (Patterns G or H), you might need a frontend. In this case, you could use plugins (Pattern A) or the Teams AI Library (Pattern F) along with it. Always consider the synergies between different patterns when planning your AI strategy.

Feedback

Was this page helpful?



Choosing the best low-code pattern to create your generative AI solution

Article • 09/20/2024

Overview

- Considerations for the low-code approach
 - Pattern selection in the ISV Journey
 - Multiple pattern option
- Explore each pattern
 - Pattern A: Create Plugins
 - Pattern B: Power Platform connectors
 - Pattern C: Graph connectors

To help Independent Software Vendors (ISVs) choose the best way to build their generative AI solution, Microsoft created guidance on the benefits of low code and pro code options.

Within this low-code journey, there is one main approach: extending a copilot. This approach is comprised of multiple patterns, which are specific paths you can take to building your solution.

This page helps you choose the best pattern for your use case if you have already chosen a low-code journey based on your needs and capabilities. If you haven't chosen a journey yet, visit the [capability envisioning page](#) to find the best approach for your use case.

Considerations for a low-code journey

The low-code approach is recommended for ISVs looking to quickly develop solutions, especially when the goal is for end users to perform work in Microsoft applications - such as Teams, Word, or Outlook. This approach typically involves limited code development, the utilization of templates, and a more rapid time-to-value than pro-code approaches.

The low-code approach may be the best option for you if you:

- *Need a quick turnaround
- *Have limited resources to devote to software development and coding professionals
- *Want to integrate your solution with Microsoft 365 productivity tools

For full customization and ongoing control, a [pro-code journey](#) might be better, even though it is more complex.

If you think the low-code approach is best for your use case, the pattern options in this article will help you to find the best tools for your needs.

There are three pattern options within the low-code approach:

- *Create plugins to enhance an existing Copilot's functionality
- *Use Microsoft Graph connectors to bring in your data to Copilot experiences
- *Use Power Platform connectors to give end users an opportunity to customize their copilot experience

Pattern selection in the ISV Journey

Within each approach, choosing a pattern is the last step an ISV takes before beginning to build a solution.

The pattern you select:

- **Impacts the capabilities of your solution.** Choosing the correct pattern for your situation enables you to match your solution to your customers' needs. Select the pattern with capabilities which align to your intended outcome
- **Affects the developmental cost of the project.** While these low code patterns deliver rapid results, some may require more lift during development. The investment of time and money needed should not outweigh the potential value to your use case
- **Enables you to work within different interfaces.** Some patterns are designed to integrate with existing applications or platforms, while others are intended to serve as building blocks for new software
- **Changes data, infrastructure, and other back-end considerations.** If you are using multiple external data sources surfaced via plugin or connectors, ensure that the plugins and connectors you choose can handle the volume of your data. Start with what's available in the [Microsoft Graph connectors gallery](#), but more data-heavy solutions might need some code changes and back-end design to meet your needs.

Multiple pattern options

Some ISVs may choose to follow multiple patterns, either to build multiple solutions or to integrate capabilities from multiple patterns into a single solution.

Each pattern offers unique capabilities to address different aspects of AI support, and utilizing multiple options can empower you to create a robust and comprehensive generative AI application. Examples of composite patterns range from data integration and automation to advanced reporting and AI insights.

Some ISVs even combine these low-code patterns with the pro-code approach to give customers multiple opportunities to engage with the software at varying altitudes. Combining both approaches can allow you to take advantage of the ease of use and quick deployment of low code along with the flexibility and power of pro-code approaches.

Whether you choose one pattern or combine options, it's important to consider the situation you are in and choose the platform that works best for you.

Pattern A: Create Plugins

ISVs looking to surface their existing services, data, and processes into Microsoft's Copilots or Microsoft 365 applications can do so by building plugins and connectors.

ISVs can create plugins using various tools, including Power Platform plugins through Copilot Studio and Teams Message Extensions . New plugins can be published to Microsoft's Copilot ecosystem via Partner Center, where IT admins can approve them for use by end users. This approach enables Microsoft 365 Copilot to interact with APIs from other software and services, surface up-to-date information, execute actions, and perform new types of computations.

You might be interested in this option if you:

- Want to bring your apps or services to Microsoft 365 with Microsoft Copilot
- Prefer to use tools like Teams Message Extension and Copilot Studio plugins
- Need to increase your solution's visibility and discoverability through Partner Center The key benefits of this pattern are:
 - Streamlining user experiences through continuity across Microsoft 365 apps, eliminating the need for users to navigate between multiple apps
 - Increasing visibility for your solution service by meeting users where they already work

Scenario

Let's imagine the hypothetical software vendor called Contoso who specialize in generative AI solutions. In this scenario, Contoso and AdventureWorks have teamed up to quickly create a solution for their virtual storefront challenges. AdventureWorks'

existing systems force employees to context switch across multiple platforms, resulting in disjointed communication and lapses in inventory management. AdventureWorks wants to improve their overall insights to promote ongoing growth.

To address these challenges, Contoso utilizes the Store Operations copilot template from Copilot Studio to create an AI assistant in AdventureWorks' existing shopping application. This plugin integrates store procedures, policies, and data into the shopping app quickly and with minimal use of resources.

Powered by AI, the assistant helps:

- Enhance employee communication by providing contextual suggestions and other relevant information from store procedures and policies in the apps where employees are already working
- Optimize inventory planning with automated alerts when stock reaches certain thresholds
- Reveal data insights by analyzing past sales performance and integrating this information into a unified view

Without having to devote significant resources, AdventureWorks was able to find a solution to their challenges. The AI assistant created by Contoso empowered them to maximize efficiency in their virtual storefront and benefit from improved insights and streamlined data.

Pattern B: Power Platform Connectors

Creating plugins is great if you want to surface your data in Microsoft 365 applications. Copilot Studio and Power Platform connectors can effectively add copilot functionality to your app. You can create Power Platform connectors in Copilot Studio to allow copilot to retrieve data from many sources including your applications. By creating these connectors, you can give your end users an opportunity to have a copilot experience based on your data and services.

You might be interested in this pattern if you:

- Have end-users interested in using Microsoft Copilot's capabilities within your application
- Require a lower coding lift but still want to create a copilot experience
- Want a copilot that can access a wide variety of data sources

The key benefits of this pattern are:

- Enriching your application with the power of existing Microsoft and non-Microsoft connectors
- Expediting plugin development through the low-code capabilities of Copilot Studio

Power Platform Connector Scenario

Contoso is partnering with AdventureWorks again because their increase in online traffic and sales last year has caused customer service challenges. They are struggling to keep up with the surge in customer inquiries and want to quickly create a specialized copilot to help manage customer service on their website.

To address these requirements efficiently, Contoso decides to equip AdventureWorks with Microsoft Copilot in their existing application using Copilot Studio and Power Platform connectors.

The customized copilot is able to:

- Integrate with AdventureWork's product database to provide real-time status updates, and assist with returns
- Connect data from their CRM, sales, and inventory to keep management informed in a streamlined manner

Since Contoso already had the data sources available, they easily created this solution using Copilot Studio and Power Platform connectors to create the right copilot experience for their client's needs.

Pattern C: Microsoft Graph connectors

Microsoft Graph Connectors facilitate the integration of external data from various sources into Microsoft 365, providing a unified and secure experience for users. These connectors enhance data accessibility, streamline development, and improve the contextual relevance of AI-generated content, leading to more powerful and integrated solutions.

You might be interested in this pattern if you:

- Need a way to integrate your enterprise application or other on-premises and SaaS cloud software with Microsoft 365 productivity tools.
- Want to enable your end-users working in Microsoft 365 to draw insights from your data sources combined with user centric Microsoft Graph data. The key benefits of this pattern are:

- Enabling the already existing Microsoft 365 client application user base to access your data and services, showcasing your offerings and potentially increasing your own user base
- Enriching insights and achieving universal integration with Microsoft 365 apps by combining ISV data with Microsoft Graph data

Microsoft Graph Connector Scenario

Contoso is working to create a solution for one of their larger clients, a multinational corporation. The client is dealing with challenges in managing and collaboration on documents across departments in offices around the globe. They use Microsoft 365 productivity applications but need a more streamlined and intelligent system for document management and to facilitate better collaboration.

To meet this need, Contoso uses Microsoft Graph connectors and Microsoft Copilot's AI capabilities to seamlessly integrate the client's various data sources into Microsoft 365 applications for centralized access and analysis.

This solution helps enhance the corporation's workflows by

- Tagging documents automatically based on context pulled from SharePoint to organize files across the global organization
- Suggesting optimal times for meetings or document reviews by integrating directly with Outlook for insights across teams and time-zones
- Utilizing Microsoft Copilot's smart summarization to help the teams quickly understand key points for improved collaboration and decision-making

By leveraging Microsoft Graph connectors for data integration and Microsoft Copilot's AI capabilities for enhanced functionality and analysis, Contoso was able to develop a comprehensive solution to the client's challenges. This solution improved their operational efficiency and gave them the tools to facilitate better teamwork and information sharing.

Conclusion

Using one of these pattern options will help you start developing your generative AI solution with low-code. If these patterns don't provide the capabilities needed for your intended use case, you may want to explore [pro-code patters](#) for more control to customize your application.

Check out these resources to learn about the tools for your chosen low-code pattern and the next steps for activation and monetization after building your generative AI

experience.

- Build message extensions for Microsoft Copilot for Microsoft 365
- Build Microsoft Graph connectors for Microsoft Copilot for Microsoft 365
- Embed a Power Virtual Agents control using chatbot control

Related links

[Creating Generative AI Experiences with the Microsoft Cloud - A Guide for ISVs | Microsoft Learn](#)
[Plugin architecture - Microsoft Copilot Studio | Microsoft Learn](#)
[Build and publish with ISV Success - Partner Center | Microsoft Learn](#)

Feedback

Was this page helpful?



Choosing the best pro-code pattern to create your generative AI solution

Article • 09/20/2024

In this article:

- [Pattern selection in the pro-code ISV Journey](#)
 - [Considerations for a pro-code journey](#)
 - [Choosing a pattern](#)
 - [Multiple pattern options](#)
- [Approach: Build a custom copilot](#)
 - [Pattern D: Microsoft Graph APIs](#)
 - [Pattern E: Azure OpenAI Assistants](#)
 - [Pattern F: Teams AI Library](#)
 - [Pattern G: Azure AI Studio](#)
- [Approach: Build an app on Fabric](#)
 - [Pattern H: Interop with Fabric](#)
 - [Pattern I: Build on Fabric](#)
 - [Pattern J: Build a Fabric Workload](#)
- [Conclusion](#)

Pattern selection in the pro-code ISV Journey

To help independent software vendors (ISVs) build their generative AI solutions, Microsoft created guidance to help you find a viable use case and start creating it. This page focuses on pro-code patterns developers can choose as they go through the journey of building their solution. If you're unsure whether to choose a pro-code or low-code pattern, visit the [capability envisioning page](#) to find the best approach for your use case.

Considerations for a pro-code journey

Choosing a pro-code approach enables ISVs like you to make use of highly customizable options when designing your AI applications. Within the pro-code approach, there are many patterns covering several platforms to address different needs and preferences. A pro-code pattern is a good fit if you are:

- Building a highly customized app and need more components within your control.
- Integrating AI capabilities into your own application or building one from scratch.

- Adhering to unique data or security concerns.

If you need a quick solution and can use tools like Power Platform connectors, consider researching [low-code patterns](#).

There are two high-level approaches within pro-code:

- **Building a custom copilot.** This approach encompasses patterns that help you to create a solution with natural language capabilities. Building with a custom copilot enables your AI to converse naturally with users in a wide range of situations.
- **Building an application on Fabric.** This approach covers patterns that use Fabric to process or store data, which can be integrated with an AI application you create. These patterns provide a strong data foundation and an opportunity for more customization.

Both approaches offer numerous benefits and enable you to create a customizable AI application. Depending on the pattern you choose, you may build an entirely new application or new capabilities that you introduce into an existing application.

Choosing a pattern

Choosing a pattern is the last step an ISV takes before beginning to build a solution. The pattern you select:

- **Impacts the capabilities of your solution.** Choosing the correct pattern for the situation enables you to align your solution to your customers' needs. Selecting a pattern with too few capabilities can limit what you can create.
- **Affects the developmental cost of the project.** Some patterns require a heavier lift during development, costing ISVs time and money. The investment needed for the pattern you select shouldn't outpace the potential value of your use case.
- **Enables you to work within different interfaces.** Some patterns are designed to create applications from scratch, while others are designed to work within your existing applications or platforms from Microsoft.
- **Changes data, infrastructure, and other back-end considerations.** Pro-code patterns are versatile but may have limits or require modifications. Typically, your data and infrastructure options become more customizable as you choose more complex pattern options.

Because of all these factors, it is essential to carefully evaluate your situation, your customers' needs, and your technical capabilities before choosing a pattern. The platform and strategy you select will impact what you can create.

Multiple pattern options

Instead of choosing just one pattern, ISVs may choose to integrate capabilities from multiple patterns. It's even possible to combine low-code and pro-code options.

Whether you choose one pattern or combine multiple, it's important to consider the situation you are in and choose the platform that works best for you. This page focuses specifically on pro-code patterns. To explore more options across pro- and low-code, you can visit the [comprehensive patterns page](#).

Semantic Kernel

ISVs looking to build sophisticated AI applications can use [Semantic Kernel](#) in many of the different pattern options. Semantic Kernel is an open-source software development kit (SDK) that makes it easy to combine your existing C#, Python, and Java code with models from OpenAI, Azure OpenAI, Hugging Face, and more.

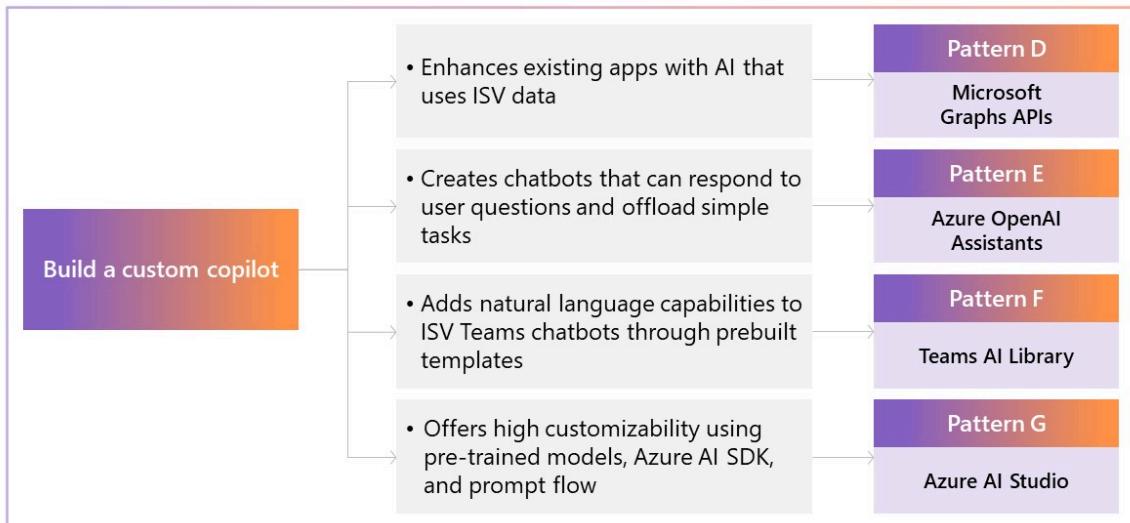
Because Semantic Kernel interacts directly with your code, it is possible to use it many different patterns. Regardless of the pattern you choose, Semantic Kernel can support your development journey and enabling your solution with new generative AI capabilities.

Build a custom copilot

Building a custom copilot enables you to create an application with a moderate coding lift and more customizability than adopting or extending a first-party, Microsoft Copilot would. While you need to build many components on your own, Microsoft provides notable support through SDKs, templates, and more, depending on the pattern you choose.

A decision tree for the "Build a custom copilot" approach. One arrow leads into a box reading "Enhances existing apps with AI that uses ISV data", which connects with Pattern D: Microsoft Graph APIs. Another arrow leads into a box reading "Creates chatbots that can respond to user questions and offload simple tasks", which leads into Pattern E: Azure OpenAI Assistants. A third arrow leads into a box that says "Adds natural language capabilities to ISV Teams chatbots through prebuilt templates", which leads to Pattern F: Teams AI Library. A final arrow leads to a box reading "Offers high customizability using pre-trained models, Azure AI SDK, and prompt flow", which leads

to



Microsoft Graph APIs

The Microsoft Graph API accesses user data from Microsoft 365 applications, such as information within Outlook, Teams, OneDrive, and SharePoint. By enabling your existing applications to call this API, you can enhance your user experience with personalized data from Microsoft 365.

These APIs can be integrated within your own solution's UI. The tenant data collected can be viewed from [Graph Explorer](#), which is an open-source platform designed to help you learn about Microsoft Graph APIs.

You might be interested in this pattern if you:

- Have an existing application you want to enhance with personalized data.
- Want to provide personalized responses to your end user based on their Microsoft 365 activity.
- Require data specifically from Microsoft 365.

The key benefits of this approach include:

- Accessing your end user's Microsoft 365 data to personalize their experience.
- Connecting to data quickly and easily, enabling you to focus on other aspects of your application.

Let's examine how one fictional ISV was able to use this pattern in their application.

Microsoft Graph APIs scenario

Contoso has built an application that enables its customers to manage internal operations, but they are seeking to improve it. Their customers bring forward issues about basic administrative tasks-such as scheduling meetings, logging time off, and sending emails-taking too much time.

To address this issue, Contoso decides to augment their generative AI application using Microsoft Graph APIs, which can connect to their customers' data in Microsoft 365. Contoso is able to enhance the capabilities of their AI assistant with access to more relevant personal data retrieved from the API. Microsoft Graph APIs enable Contoso's solution to:

- Generate calendar entries and time away emails based on context from the user and their time off requests.
- Use information from their end customers' Outlook calendars to suggest potential meeting times and invitees.
- Suggest tone edits, subject lines, and documents to attach to emails based on history between recipient and sender.

These changes and more enable Contoso's generative AI application to dramatically streamline their customers' administrative tasks. By using Microsoft Graph APIs in their application, they could provide useful and personalized advice for employees.

Azure OpenAI Assistants

Using the capabilities of [Azure OpenAI Assistants](#), ISVs can quickly create AI assistants and integrate them into their existing applications. Azure OpenAI Assistants can answer questions, prompt simple tasks, and even be adapted to write and execute code based on a user's inputs.

Creating an Azure OpenAI Assistant is as simple as writing a JSON file describing the function you would like the assistant to perform and giving it a sandboxed Python environment in which to run. This enables the assistant to call your existing APIs and begin responding to prompts.

You might be interested in this pattern if you:

- Have an existing application that would benefit from a customized assistant.
- Want to develop a copilot-like application quickly with less technical lift.
- Need to integrate with additional tools that enable your application to complete tasks without language, such as math.

The key benefits of this approach include:

- Creating an AI assistant quickly and efficiently.
- Providing your customers with an AI assistant capable of answering more specific questions and requests.
- Enabling your solution with AI capabilities that can prompt simple actions to streamline tasks.

Azure OpenAI assistant scenario

Contoso has an existing application for retail customers that provides an employee portal for onboarding, inventory management, payment processing, and more. Contoso will use Azure OpenAI Assistant to create a store assistant, providing customers with guidance tailored to their specific situation.

By creating an AI assistant, Contoso can integrate the businesses' data into the application, which allows it to answer questions using company data. These capabilities allow the copilot to:

- Guide new employees through typical store processes.
- Predict inventory needs and requirements based on past trends.
- Reference guidance in uploaded employment documents, such as leave policies.

By including an AI assistant within their application, Contoso's solution becomes a source of guidance and suggestions for employees, rather than just a management tool. Their customers can use it to answer questions and identify a path forward in day-to-day tasks.

Teams AI library

If you already have a Teams chatbot or are interested in creating one, your chatbot can be enhanced with generative AI capabilities. Scaffolding within [Teams AI library](#) is able to support conversational language for your chatbot, which can be accessed directly by users in Teams.

This application requires you to input the business logic you would like your app to use while the large language models (LLMs) that Microsoft provides handle the back-end aspects of the chatbot. You can even modify your chatbot to use different LLMs, plugins, and more.

You might be interested in this pattern if you:

- Have an existing Teams chatbot that you would like to augment with natural language capabilities.
- Want to leverage pre-built templates, integrated data options, and built-in safety features.

The key benefits of this approach include:

- Improving your Teams chatbot with natural-language capabilities.
- Building a generative AI solution with ample support options.
- Customizing your chatbot to apply to your situational or industrial needs.

Teams AI library scenario

Contoso has made Teams chatbots available to their customers for some time now, but as AI capabilities evolve, they want to modernize their application by enabling it to answer more specific questions and provide personalized advice to the employee.

Using the Microsoft Teams AI library, Contoso can easily add natural language capabilities to their chatbot. These new generative AI capabilities enable the chatbot to better support customers by:

- Suggesting tactical next steps for ongoing projects and tips to prepare for upcoming company events.
- Creating meeting agendas, email drafts, and more based on a short interaction between the user and chatbot.
- Answering questions using tailored responses based on the context of the conversation.

These features help streamline the employee experience for Contoso's customers and enables them to get more intelligent answers, faster. Incorporating generative AI into their Teams chatbot makes its users more efficient at work, as they can naturally converse with the chatbot to receive support at work.

Azure AI Studio

If you want to create a completely customized generative AI application that is highly flexible in terms of capabilities, you can use one of the many cutting-edge options from [Azure AI Studio](#). Building generative AI solutions with Azure AI Studio enables you to customize your solution to your specific needs, including meeting highly technical or niche requirements.

Azure AI Studio includes many options to build an AI application, such as prompt flow and the Azure AI SDK, both of which have pretrained models from which you can build.

The service is integrated with other Azure services and provides resources for ongoing application development, such as an LLMOps toolchain.

You might be interested in this pattern if you:

- Want to build an application from scratch or modify an existing one.
- Require your AI assistant to complete complex processes, such as reading and reviewing technical documents or analyzing complex data trends.
- Want full control over app development to customize voice, personality, and brand identity to your specific needs.

The key benefits of this approach include:

- Creating a bespoke copilot that can respond to more niche questions and requests.
- Customizing your solution to adhere to unique security or data requirements.

Azure AI Studio scenario

Contoso wants to build a fully customized generative AI application to support their healthcare customers during claims creation and processing. To commercialize the application, they needed to be able to accurately process customer information, meet specialized security considerations, and generate accurate content.

By using Azure AI Studio to create their own generative AI application from scratch, they were able to build a fully customized, highly secure application tailored to the needs of their healthcare customers. Contoso's new application enables clinicians working for insurers to easily fill out forms. The copilot they created through Azure AI Studio can:

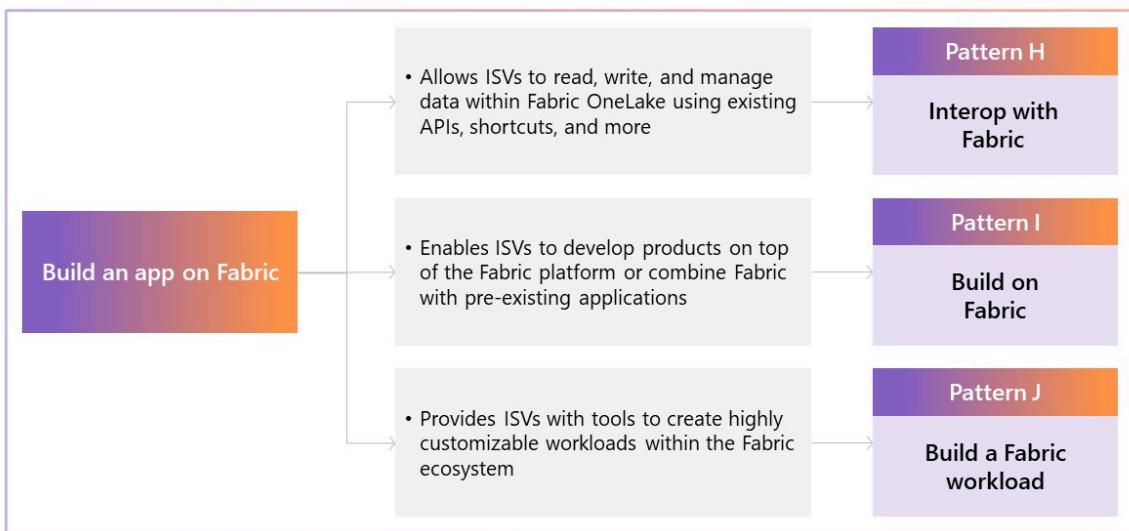
- Support building conditional authorization forms through conversational prompting based on a variety of organization-specific questions.
- Query multiple patient records and review information within Contoso's application.
- Meet organization-specific security considerations to minimize risk and privacy concerns.

By accurately retrieving patient information and using natural language capabilities to assist in generating forms, Contoso can accelerate the claims creation process. Their solution saves users valuable time by completing these specialized tasks quickly and accurately.

Build an app on Fabric

While many ISVs are familiar with Fabric as a data or analytics solution, it can also serve as a comprehensive, integrated data foundation for generative AI applications. Whether you want to build an app directly on Fabric or integrate with OneLake, [building an app on Fabric](#) provides you with unmatched customization and control over your solution.

A link tree for the "build an app on Fabric" approach. One arrow leads to a box reading "Allows ISVs to read, write and manage data within Fabric OneLak using existing APIs, shortcuts, and more", which leads to Pattern H: Interop with Fabric. A second arrow leads to a box reading "Enables ISVs to develop products on top of the Fabric platform or compine Fabric with pre-existing applications", which leads to Pattern I: Build on Fabric. A final arrow leads to a box reading "provies ISVs with tools to create highly customizable workloads within the Fabric ecosystem", which leads to



Interop with Fabric

Integrating your existing application with Fabric provides endless options for streamlining the data backend of your solution. By interoperating with Fabric, you can unify a multitude of disparate data sources on a single platform using a variety of tools and APIs.

Fabric offers both OneLake APIs and Real-Time Intelligence APIs that are designed to quickly access and process your data. You can also use Data Factory to unify your data across environments through shortcuts and manage complex data processing tasks from up to 200 external sources. Even if your data isn't on Azure, you can create shortcuts to bring your data into OneLake.

You might be interested in this pattern if you:

- Have an existing application that requires enhanced data processing.
- Are drawing your data from a variety of sources and environments.
- Want to read and manage your application's data within OneLake or Data Factory.

The key benefits of this approach include:

- Improving data capabilities for existing apps through increased processing, storage, and analytic abilities.
- Integrating data from multiple tenants and environments into one platform.

Interop with Fabric scenario

The SaaS application that Contoso has developed for their retail customers requires greater data capabilities to manage data from multiple tenants. Contoso's end customers require greater visibility into their data and are requesting more information about trends and patterns within their data. However, their customers use a variety of platforms for inventory management, staffing, web management, and more that Contoso needs to access to form a complete view of the data.

To address these issues, Contoso connected their existing application to Fabric. This allows them to have better data management at an administrative level, as well as better data insights and visibility for tenants. Their application is able to:

- Use data sharing and multi-cloud shortcuts to gather information from different tenants in OneLake.
- Access organization data immediately through Real-Time Intelligence APIs.
- Handle amounts of data associated with a large-scale solution.
- Portray information about customer's data to them through interactive dashboards that Contoso created in their application.

Now that Contoso can access data from their customers more easily, they are able to modify their solution to provide that information to their end customers. Their end customers can easily and accurately interpret their data and use it to make choices for their business.

Build on Fabric

Beyond connecting to Fabric as a data platform for your application, you can also build on Fabric to embed Fabric's functionalities directly into your app. Developers can use a variety of REST APIs to build Fabric capabilities into their applications to support more technical workflows with generative AI capabilities.

Different APIs can be integrated within your application to enable it with different functions. For instance, embedding the Warehouse API within your application provides it with the Data Warehouse options available within Fabric. Developing your application directly on top of Fabric allows you to seamlessly utilize and manipulate these Fabric functions within your own application's interface.

You might be interested in this pattern if you:

- Are building an application that will be used by data scientists or other users with more comprehensive data management needs.
- Want to incorporate comprehensive data storage and processing options in your application.
- Are looking to build an application from scratch or modify an existing one.

The key benefits of this approach include:

- Integrating Fabric's data capabilities directly into your solution.
- Managing and manipulating data within your own application.
- Creating your application with full customizability and control.

Build on Fabric scenario

Contoso's customers have been expressing a need for more complex data capabilities. Between tracking sales trends, employee schedules, inventory for digital and in-person stores, and more across multiple tenants, they require much more data processing support in their new application.

To more effectively manage customer data, Contoso developed a new application built on top of Fabric. This application allows users to interface directly with their data and interact with it in Contoso's application. By harnessing Fabric's data processing capabilities, their application can:

- Connect to OneLake, Power BI, and more directly within the application, which can be surfaced through customized dashboards built by Contoso.
- Provide data insights into profitability from categories such as online or in-person stores, different store locations, and by product brand and category.
- Use AI to analyze this data and provide insights and suggestions to customers, such as suggesting a change in pricing for a product, predicting potential disruptions, and identifying outliers in revenue streams.
- Provide Contoso's customers direct access to their Fabric data within Contoso's platform.

By embedding Fabric's capabilities within their application, they can manage and manipulate data directly within their solution. The AI they've built on top of their solution is able to access this data and provide personalized business insights for Contoso's customers.

Build a Fabric workload

In addition to building with Fabric's seven native workloads, ISVs can expand Fabric's capabilities by creating their own customized workloads and offering them as a separate solution. These workloads can be built from scratch to deliver a plethora of data management capabilities, whether that's creating a more wholistic view of your customers data or taking actions based on data trends and predictions.

ISVs can use the [Microsoft Fabric Workload Development Kit](#) to create their own workload and publish it as a SaaS offering for other Fabric users on the Azure Marketplace. This pattern is easily monetizable on Azure Marketplace and uses Fabric UX when interfacing with customers, which leaves you free to focus on developing your workload.

You might be interested in this pattern if you:

- Would like to create a tool available to end-users on Fabric, such as other developers or data scientists.
- See a data need or requirement that you want to address with Fabric.
- Want to leverage Microsoft support like existing UX and simple publication on Azure Marketplace.

The key benefits of this approach include:

- Publishing your application in Azure Marketplace, where it can be easily accessed and purchased by technical end users in Fabric.
- Creating highly customizable solutions directly tailored to your customers' data requirements.

Build a Fabric workload scenario

Contoso wants to create solutions that can be used by their retail customers to track supply chains and inventory management across both digital and in-person stores for several tenants. They want their application to be easily accessible for technical end users so that they can pursue more use for their data-driven application.

Using the Microsoft Fabric Workload Development Kit, Contoso was able to develop a workload their customers can access directly from Fabric in a UX they are familiar with.

They were able to monetize it as a SaaS application in Azure Marketplace, where it is easily accessible to Contoso's end users in the Fabric platform. Their Fabric workload is able to:

- Optimize inventory by providing insight into areas that need additional demand and using AI capabilities to suggest a path forward.
- Forecast future demand through machine learning capabilities that analyze past trends.
- Simulate possible scenarios that could affect the supply chain, such as changing suppliers.

By offering their solution as a Fabric workload, Contoso can help data scientists and other technical professionals optimize supply chains. Contoso's customers gain greater visibility into their business, through the AI's careful analysis of past data and predictions of future trends.

Conclusion

By learning about each pattern and their capabilities, you should now be equipped to decide how to build your generative AI solution. After researching your chosen approach and confirming that it is within your capabilities, you can begin developing your application.

Explore the resources below to learn more about your chosen pattern as well as other next steps to building your generative AI experience.

Build your own copilot

Links to more information for each build your own copilot pattern:

- [Pattern D - Microsoft Graph APIs](#)
- [Pattern E - Azure OpenAI Assistants](#)
- [Pattern F - Teams AI Library](#)
- [Pattern G - Azure AI Studio](#)

More information about building an app on Fabric

Links to more information about building apps on Fabric:

- [Approach - Build an app on Fabric](#)
- [Pattern J - Build a workload on Fabric](#)

Related links

More information on generative AI patterns and their benefits: [Creating Generative AI Experiences with the Microsoft Cloud - A Guide for ISVs | Microsoft Learn](#)

UX guidance for ISVs designing generative AI experiences: [Next steps for designing your generative AI user experience.](#)

Feedback

Was this page helpful?



Creating a dynamic UX: guidance for generative AI applications

Article • 09/20/2024

So you proposed a use case for your AI feature or application, justified it with a business case, and received approval. Great job! Now what? Well, now your stakeholders are ready to see something.

When planning to build an application with generative AI-based features, it's important to consider the tips and guidance provided here to create the user experience (UX) you want. Understanding these key principles can help ensure you're building an engaging, effective application that best supports the needs of your users.

The following guidance walks you through the considerations to have in mind when developing the UX for your generative AI application, with a focus on considerations for building custom copilot experiences.

This article is intended to help you:

- Explore UX framework options and select the best approach for your use cases.
- Learn three foundational principles for developing a copilot, and guidelines for human-AI interaction.
- Discover how to achieve a collaborative UX through strong input and output design

The following guidance will walk you through the considerations to have in mind when developing the UX for your generative AI application, with a focus on considerations for building custom copilot experiences.

This area is very much in flux and there's a lot to learn, so we've provided key resources for deeper diving. [Microsoft's responsible AI principles](#) and the [HAX \(HumanAI Experience\) Toolkit](#) give some background on the research and real-world experiences behind this article.

The right focus to get the job done

The following guidance will walk you through the considerations to have in mind when developing the UX for your generative AI application, with a focus on considerations for building custom copilot experiences.

There are three framework variations to consider for your UX:

- Immersive for a whole knowledge base focus
- Assistive in-app focus
- Embedded for single entity focus

Let's explore each of these frameworks in more detail

Immersive focus for a whole knowledge base

A good rule to follow is the more important the task, the more real estate required.

An immersive environment provides a fully focused experience by utilizing the entire canvas to display relevant information, allowing for deeper insights and reducing distractions for the user. This level of focus is ideal for applications where you want to display information that is related to specific data sources. Examples of this include AI-generated dashboard similar to [Microsoft's Project Sophia](#), or the way in which [Microsoft Copilot for Security](#) guides users through a comprehensive process. In an immersive space, complex data or information becomes easier for users to understand and analyze.

Assistive focus for an in-app experience

Give your users the ability to access AI powered assistance from within the applications that they already work in - like Microsoft Teams, Power BI, or your own apps - by integrating copilot as an assistant to extend existing functionality.

By using in-app focus, users can avoid switching between tools or interfaces. This framework allows the copilot to seamlessly integrate into the user's workflow, providing relevant suggestions, information, and support on demand without disrupting their current task. This view provides continuous access to tools, information, and assistance without obstructing the main content area. It is particularly effective for applications that require ongoing support or monitoring.

Embedded focus for a single entity

Embedding a single-entry point can simplify the integration of a copilot into your application, reducing complexity and allowing users to receive support on specific items or actions. This helps create a seamless copilot experience with context-aware assistance without occupying permanent screen space.

This option can be ideal for tasks that require only occasional guidance or interaction, though it might not be suitable for more complex or detailed interactions. Fully embedding copilot should align with common interaction patterns, such as highlighting

a portion of code to invoke copilot to take action, or enabling users to dive deeper into a chart on an analytics dashboard.

Incorporate a secondary focus

In addition to using any of these three frameworks individually, you can create a more robust experience by complimenting your chosen focus with an additional framework option. We find incorporating an embedded option with an immersive or assistive copilot can provide further value for users.

Regardless of what level of focus you choose for your use case, your ultimate goal should be to give your user a copilot experience is positive and productive. The following guidelines are intended to help you maximize the success of your copilot through effective UX design.

Three Foundational principles for copilot UX

AI driven experiences can be impressive and it's not uncommon for people to have an emotional, trusting response to what seems like conversational, original content. But a copilot simply uses the information it was trained on to predict a word-by-word response with no inherent understanding of truth. Therefore, it's important that you base your copilot on the following principles, setting appropriate expectations for your users.

Principle 1: Human in control

All great copilot experiences are grounded in the following basic concept: a copilot is simply a tool to support the user. The human is the pilot.

To set this expectation, position the user in the driver's seat. This means giving them the information they need while still providing transparency around how the copilot works. Communicate its abilities and limitations, and give clarity to the data that outputs are based on. Package this information in meaningful human controls to enable users to confidently and iteratively guide the copilot towards their goals.

For example, when introducing a copilot feature, don't lock up the word "copilot" with action words on the UI. Instead of "copilot, summarize," say "Summarize with copilot." This language reminds the user that the copilot is simply an assistant.

Principle 2: Avoid anthropomorphizing copilot

Many of the generative AI experiences available today can closely imitate natural human language. Because the technology is so good at doing this, there's potential for users to develop inappropriately high expectations about its nature and abilities and therefore over rely on the copilot's responses.

There are a couple ways you can help prevent users from making these assumptions:

1. **Give copilot its voice.** To avoid the perception that copilot is human-like, teach it to use the right language and avoid certain words in its responses. For example, avoid words like "understand," "think," or "feel" in any context as they may convey copilot is human-like. Instead, use words that relate to machines like "processing" and analyzing."

However, allowing copilot to use first-person singular pronouns (I, me, mine, myself) in its responses works well because it is more conversational. And although using first-person plural pronouns (we, us, ours) to refer to your user and your copilot together is fine, don't use those pronouns to solely represent your company. Why? Because that gives copilot license to be your voice, and in some cases could appear to be speaking on behalf of your company.

2. **Go light on personality.** Consider the implications of what you call your copilot within the UI. How do you introduce it to your users, and how do you mention it in your marketing and support materials? The more character you give it, the more you humanize it.

Principle 3: Consider direct and indirect stakeholders

Like all technology, generative AI applications have impact that can reach beyond just the primary user. Throughout the design process, consider not just the immediate user but everyone the product might impact, especially the most vulnerable direct and indirect stakeholders. Make it a habit to design for both primary users and anyone else who might see the output. It's important to consider the wider consequences, including the possible unintended consequences, of your generative AI applications.

These considerations are different for every organization, so discuss it with your team and a few potential users, asking questions like:

- How might this output be used?
- Are users sharing it with anyone else?
- Should other teams or groups review our generative AI strategy?
- Who are the most vulnerable stakeholders and how might we protect them?
- Are we implementing meaningful human controls to empower users with different abilities?

- What might be the unintended consequences if the technology fails or if it's misused?

Designing experiences for the application lifecycle

First run experience

When your users first invoke your copilot, they should find it engaging enough to start a conversation. They should feel confident in what copilot can and can't do, so be sure to show new users the different ways in which they can use the AI.

Microsoft studies show that users prefer an experience that explains what the copilot can do and gives them suggestions on how to begin. There are many ways to create such an experience and we encourage you to try different approaches with your end users. The following considerations come from the [HAX toolkit](#), and have design patterns available which offer several techniques you can mix and match to set user expectations:

- **Make clear what the system can do.** Help the user understand what the AI system is capable of doing.
- **Make clear how well the system can do what it can do.** Help the user understand how often the AI system may make mistakes.

Remember, everyone's learning in this space, and to succeed in any copilot endeavor, you need to keep an open mind, and think creatively. Be ready to experiment, learn, discover, and even conduct your own research.

Other guidelines should be considered over the course of the application's lifecycle. Some of the most relevant are listed here with links to their corresponding patterns. Learn more about Microsoft's other guidelines in the [HAX Design Library](#).

During interaction

- **Match relevant social norms.** Ensure the experience is delivered in a way that users would expect, given their social and cultural context.
- **Mitigate social biases.** Ensure the AI system's language and behaviors don't reinforce undesirable and unfair stereotypes and biases.

When it's wrong

- **Support efficient correction.** Make it easy to edit, refine, or recover when the AI system is wrong.
- **Make it clear why the system did what it did.** Enable the user to access an explanation of why the AI system behaved as it did.

Over time

- **Encourage granular feedback.** Enable the user to provide feedback indicating their preferences during regular interaction with the AI system.
- **Provide global controls.** Allow the user to globally customize what the AI system monitors and how it behaves.

Collaborative UX

Copilots can improve existing information by making changes or creating new examples without needing extra data. However, this capability also means that a copilot might sometimes generate wrong or unhelpful responses.

To reduce the likelihood of fabrications, one good practice is to enable your users to guide the copilot and move it towards their personal goals and objectives in what is called **collaborative UX**.

You can create a collaborative environment for the user with the following tips for input and output design. You may also find it helpful to align with these [best practices for building collaborative UX with human-AI partnership](#)

Tips for input design

Effective input design forms the cornerstone of the collaborative experience. By guiding users to make well-structured inputs, It lays the foundation for relevant and accurate responses.

1: Provide Suggestions to help users get going

Because generative AI is new tech, it's hard for many people to know what to do or type right away. Long-form natural language typing is still not a habit for many. To help users get going, offer clear suggestions and affordances like large input boxes and character counters that encourage them to form good inputs in addition to a pleasant onboarding experience.

For more specific needs, adding features like promptbooks to give the user specific, short queries that interact with custom data in predictable and repeatable ways may serve to yield useful information faster.

2: Encourage details

Another way to help users create good, detailed inputs is by designing an experience that uses a variety of elements.

For example, you can separate one general prompt into multiple input fields. Replace the question, "What would you like to blog about?" with four inputs, such as:

- Type a title
- Add some more details
- Include images
- Describe the tone

3: Allow customization of inputs using tone and other options

Speaking of tone, help users customize their inputs by providing predefined options at the start. Make sure the tone settings are clear to your users and let them know they can change tone settings anytime within a conversation.

4: Enhance user interaction and engagement with multimodal design

To enable users to effectively engage with a copilot through whatever device or method they prefer, offer multiple modalities in your input interface. This effort for inclusivity can mean adding both voice and text options and extends to allowing for multi-lingual inputs. Giving the user multiple options to create inputs enables them to more easily and collaboratively communicate.

Tips for output design

In a collaborative UX approach, your users need to guide the copilot with a continuous feedback loop between inputs and outputs to reach their goals. Output design creates avenues for the user to shape and influence the copilot's responses and drive towards the desired output.

1: Show inputs and outputs together

This helps your users associate output quality with input choice, affording a tight feedback loop where users can continue to build inputs until the model produces the outputs they desire.

2: Keep a history of outputs and prompts

Encouraging users to try various inputs to get to meaningful outputs is crucial; however, it is not always a forward interaction with outputs getting better each time.

Sometimes a new prompt may lead to a worse output. A timeline or history of outputs allow users to try new inputs confidently without fear of losing access to previous outputs that may be better, or even use parts of multiple outputs.

Likewise, allowing users to make use of previous prompts is deeply valuable to the iterative process.

3: Add appropriate friction (it's a good thing!)

We often want to remove friction from product experiences. But remember that copilot is an imprecise ("probabilistic") system that is likely to make mistakes. Because of that possibility, you need to add appropriate friction to help users build a new mental mode.

The goal here is to slow down users and encourage them to review outputs throughout. Add friction at key moments like save, share, copy, and paste, and make clear to the user that they're about to take ownership of the content. Therefore, they take accountability for content that they use encouraging them to check it more thoroughly first.

Here it's recommended that you encourage users to edit content to provide more context or add a personal touch to it. Add AI notices and disclaimers with each output that clearly express AI-generated content may be incorrect.

4: Encourage fact-checking using citations and direct quotes

One specific way to encourage fact-checking is to have a copilot show references from the data it's referencing, making the AI more likely to use responses from existing resources instead of fabricating data and information. These references also remind users to take accountability for the content they use by having a second look at copilot's output and verify it against its sources.

By integrating direct quotes from the source and directing the user to the specific location of that information, your copilot can support more thorough fact-checking. These quotes help copilot to remain aligned with the training dataset and build suitable trust and appropriate reliance.

A final note on fact-checking: showing references doesn't completely prevent a copilot from making things up. Go one step further and design an experience that slows the user down (see "Add appropriate friction") and encourages them to review responses.

5: Allow user to edit outputs

A copilot can come close to a desired output but may not exactly match it. You may find that some context is missing. Perhaps the response sounds too general or doesn't match your usual personal tone as opposed to if you created similar content yourself.

A key part of collaborative UX is letting the user intervene and modify the output. It also shows that copilot is a helper or assistant with the user as the pilot.

6: Withhold outputs when necessary**

In some cases, it's better for a copilot to give no answer instead of outputting something potentially inappropriate. You may want the model to disengage and prompt the user to start a new chat with "Sorry, I can't chat about this topic. To save the chat and start a fresh one, select New Chat."

Sometimes there are error states and inappropriate inputs that necessitate you to create prewritten responses. For harmful or controversial topics like self-harm and elections, Microsoft recommends that the copilot not disengage but instead use predefined experiences. In cases where you don't want to disengage the user fully but simply want to redirect the conversation you could recommend that they "please try a different topic."

Every case is different, and it's best to tailor these responses to your application's purpose and expected use.

7: Allow users to provide feedback about the output

Design mechanisms for users to evaluate your copilot's outputs through things like accuracy rating systems, options for users to ask the copilot to correct responses, mark responses as helpful or unhelpful, or leave comments on received outputs. You can also demonstrate how user feedback is helping improve their copilot outputs and experience to reinforce the value of their feedback.

Next steps

Now that you know how to achieve your desired user experience, here are Microsoft's resources and tools to help you as you begin building your generative AI application.

- [Microsoft HAX Toolkit ↗](#)
 - [UX for AI: Design Practices for AI Developers | LinkedIn Learning ↗](#)
 - [Learn how to use Microsoft Copilot | Microsoft Learn](#)
 - [Responsible AI Principles and Approach | Microsoft AI ↗](#)
 - [Accessibility - Fluent 2 Design System ↗](#)
-

Feedback

Was this page helpful?

 Yes

 No

Observability at scale for ISV Solutions on the Microsoft Cloud

Article • 09/20/2024

If you're an independent software vendor (ISV) building applications on the Microsoft Cloud, you likely require insight into your applications' data after deployment. Ensuring visibility into the performance of your application-no matter where it's running-is critical to maintaining and improving it. If you've deployed large-scale solutions on Azure, you might know the [Azure Monitor Metrics Data Plane API](#). It helps you query metrics across subscriptions and meet your observability needs.

When deploying a Microsoft Cloud application, there are several [landing zones and deployment models](#) ISVs can choose from. The ways that you practice observability varies depending on the deployment pattern and platforms you choose. For instance, it's typically easier for an ISV to monitor their own SaaS platform than to gain visibility into software that a customer deploys themselves.

Observability challenges and considerations

When a solution is developed using Microsoft Cloud capabilities across platforms such as Azure, Teams, Fabric, and more, the challenge becomes effectively monitoring large deployments at scale. Monitoring large-scale AI application deployments requires you, as a ISV, to aggregate metrics from various cloud subscriptions, identify and isolate issues, and then begin the troubleshooting process.

Some of the considerations that ISVs can address through effective monitoring include:

- Evaluating core infrastructure availability for potential issues
- Identifying a need for scalability for their applications
- Processing data quickly and effectively
- Efficiently responding to errors

If you're building generative AI applications, you may have even more challenges and considerations for observability, as AI introduces complexity into topics like data regulation, ethical use, and privacy issues. For information on metrics and telemetry considerations specific to generative AI, visit this guide on [Observability for generative AI](#).

Enabling observability at scale through Azure Monitor

The [Azure Monitor Metrics Data Plane API](#) can improve resource insight gathering by unlocking higher querying capacity and increased efficiency for ISVs like you.

This API can, with a single batch call, retrieve up to 50 resource IDs, gather customer information, and provide essential clues for effective troubleshooting. These capabilities help you improve query throughput, manage throttling risks, and improve your experience as you gather insight into your solution.

The API requires code modifications by the ISV for some deployment methods but is ready to use for certain pure SaaS applications.

Gathering metrics through Azure Monitor

Through APIs like the Azure Monitor Metrics Data Plane API, ISVs are able to query essential information from various sources in real time.

Metrics supported

Through built-in querying capabilities and options for revision and customization, the API is able to provide ISVs with various metrics. It's capable of retrieving Azure resource metrics. Some of the metrics you can use the API to retrieve include:

- Firewall health
- Storage availability
- Log analytics rates
- Load balancer availability
- Azure Kubernetes Service (AKS) Server Node status
- Key vault availability
- Container registry success
- Cosmos DB Availability

Sources

Azure Monitor can gather data from a wide range of sources and application types, whether they are hosted in Azure, other cloud environments, or on-premises.

You can use the API to gather data from sources such as:

- Apps

- Workloads
- Databases
- Infrastructure
- Guest operating systems
- Azure Platform
- Custom Sources integrating with Azure Monitor APIs

Data storage

Azure monitor is capable of retaining information in data stores. If you're looking to archive data for a longer amount of time, you can export your data to Azure Storage.

Consumption

Consuming your data in useful ways is the most important phase of observability. Azure Monitor is equipped to help you visualize, analyze, respond to, and gather insights into your data.

[] [Expand table](#)

Azure Monitor Capability	Tools and Options
Insights	Azure Monitor is equipped with many insights into applications, containers, networks, and more. These insights can provide comprehensive information for the performance and health of everything from your Kubernetes clusters to your Linux VMs.
Visualize	Helpful workbooks and dashboards allow you to turn data into charts, tables, and graphs that can be shared across teams. Grafana and Power BI are also integrated into the Azure Monitor portal.
Analyze	Through tools like metric explorer, log analytics, and change analysis you can query for trends and issues in your metric values, log data, and resource changes respectively.
Respond	Azure Monitor has the capability to not only alert you of critical conditions, but to act on issues as well. Azure Monitor's capabilities can be further enhanced with AI services that automate tasks and autoscaling features that manage resource loads efficiently.

Presentation

You can use tools like Grafana for easy visibility into your solution's health and performance. The dashboard is customizable for ISVs who want to display extra metrics.

Observability based on deployment patterns

Pure SaaS deployments

If you're deploying your application as a pure SaaS subscription, practicing observability is easily enabled within your own infrastructure. While you are able to query data directly, it is important to maintain consistent supervision over your environment to ensure it is able to perform well.

Key observability considerations for pure SaaS deployments

- Applications deployed into your own Azure subscription can use the Azure Monitor Application Insights capabilities.
- You can also use the Azure Monitor Metrics Data Plane API to query metrics at scale.

Customer-deployed

When your customer deploys the solution within their own environment, they may be able to see benefits such as greater control and security. You can provide them with observability solutions, encourage them to build their own, or obtain customer consent to gather query data from external sources.

Key observability considerations for customer-deployed applications

- Your customers use an observability solution that you provide to them or they can build their own observability solutions in situations that may require unique integration, privacy, or security considerations.
- Custom-built solutions may allow you to capture telemetry and metrics from customers' subscriptions and enable metric data transfer after your customer consents.

Dual-deployment SaaS

This deployment pattern allows some pieces of the component to run within the customer's subscription, while others run within your infrastructure. For example, you

could run the backend services required for processing data in Azure while the components from D365 may run in your customer's tenant.

Key observability considerations for dual-deployment SaaS applications

- You can use Azure Monitor to oversee any infrastructure within your own subscription, but you won't have visibility into components deployed in your customers' subscriptions.
- You may obtain other data from consenting customers by using custom solutions similar to the customer-deployed model.

Next steps

Given these deployment scenarios, you can use the [MCSA-Observability](#) solution accelerator to kick start your observability journey. This solution accelerator utilizes the capabilities of the Azure Monitor Metrics Data Place API solution and is available to all ISVs for use in monitoring your solutions at scale.

Related links

- [Azure Monitor overview - Azure Monitor | Microsoft Learn](#) - In-depth guidance on Azure monitor and its many capabilities.
- [Independent software vendor \(ISV\) considerations for Azure landing zones - Cloud Adoption Framework | Microsoft Learn](#) - An ISV guide to different landing zones within Azure.
- [Azure Monitor- Announcing General Availability of Azure Monitor Metrics Data Plane API - Microsoft Community Hub](#) - A quick introduction to the Azure Monitor Metrics Data Plane API.
- [Observability at scale - Azure Monitor Metrics Data Plane API - Microsoft Community Hub](#) - An in-depth review of the Azure Monitor Metrics Data Plane API and how it related to Observability at scale.
- [Metrics Batch - Batch - REST API \(Azure Monitor\) | Microsoft Learn](#) - Specific information on the Azure Monitor Metrics Data Plane API, such as URI Parameters, request bodies, and more.

Feedback

Was this page helpful?

 Yes

 No

Observability for pro-code generative AI solutions

Article • 09/20/2024

Overview

- Building and operationalizing copilot applications
- Observability challenges in generative AI
- Metrics to monitor and evaluate
- Developer tools to get started
- Next Steps

Introduction

Microsoft partners, including independent software vendors (ISVs) like yourself, can create generative AI solutions through many different pro-code and low-code approaches. To support you in this process, Microsoft is [creating guidance](#) for ISVs to better enable you to build these solutions.

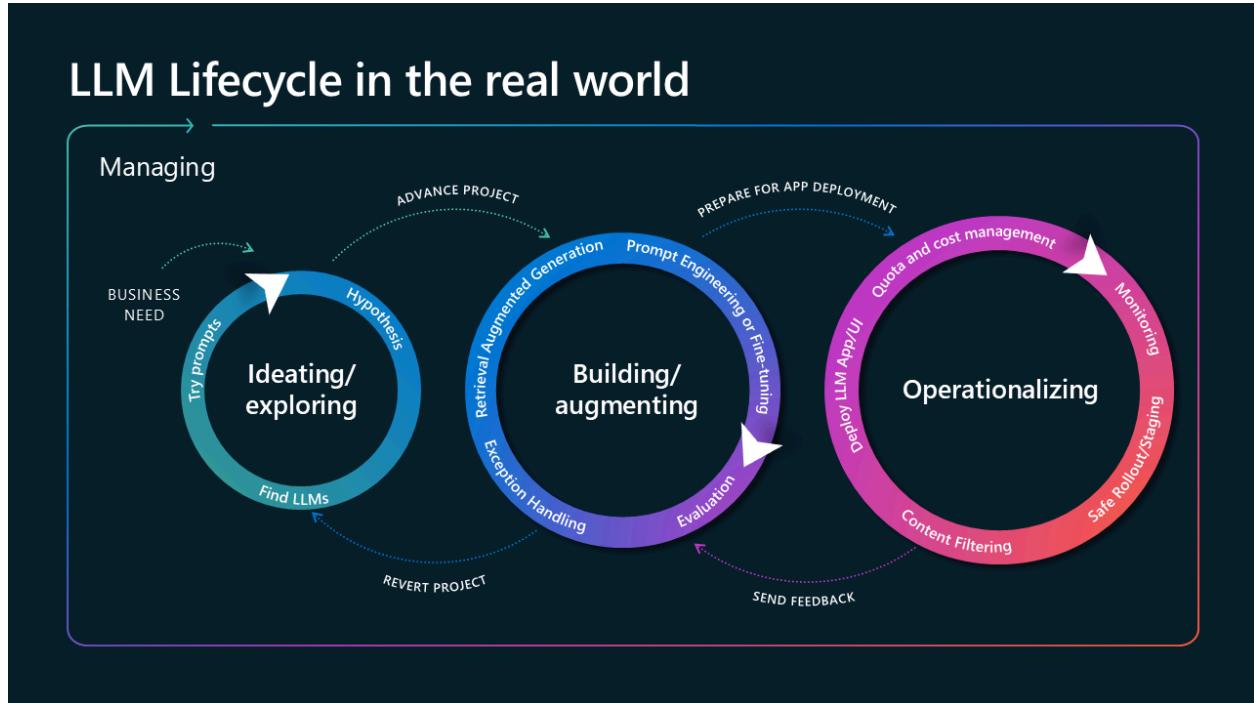
As ISVs aim to manage specialized queries and tasks, the complexity of their generative AI solutions increases. These complex generative AI solutions require unique precautions during development and consistent monitoring and observation throughout production. By observing your product's behavior and outputs, you can quickly identify areas for growth, promptly address risks and issues, and drive even higher performance for your application.

Building and operationalizing copilot applications

To understand how observability impacts your application from the beginning of the solution lifecycle, it's key to think of the lifecycle in three major stages: ideating on your use case, building your solution, and operationalizing it for use after deployment.

Image titled LLM Lifecycle in the real world. It's composed of three circles, connected with arrows and surrounded by a larger arrow labeled "managing". The first circle is labeled "Ideating/Exploring" and includes "try prompts", "Hypothesis", and "Find LLMs". It stems from the business need arrow and is connected to the second circle with an arrow labeled "advance project". The second circle is labeled "Building/augmenting" and

is composed of "retrieval augmented generation", "exception handling", "prompt engineering or fine-tuning", and "evaluation". The second circle connects to the last circle with "prepare for app deployment". The last circle is the largest, is labeled "operationalizing", and is composed of "deploy LLM app/UI", "quota and cost management", "content filtering", "safe rollout/staging", and "monitoring". The last circle is connected to the second circle with "send feedback", while the second is connected to the first with



The first phase is made up of identifying your use case and ideating on technological approaches to building it. Once you identify a path to building your application, you enter the second phase, which consists of developing and evaluating the application. After the application is deployed to production, it enters the last phase, where it can be observed and updated.

Observability insights gleaned from the second and third phases become critical when returning to earlier phases to perform updates and deployments. By performing continuous testing and retrieving metrics to inform earlier processes, you are able to keep fine-tuning your application.

It's important to understand what actions you can take to practice observability at different points in the process. At a high level, the following observability actions happen during the application lifecycle:

First phase

1. Non-technical personas such as product managers practice observability by ideating on key qualities of their application.

- Stakeholders can define what metrics are most important to measure the performance of their application, such as risk and safety metrics, quality metrics, or executional metrics.
 - Teams can define goals for metrics such as user engagement and cost management.
2. Technical personas focus on identifying platforms, tools, and methods that can most successfully enable building the application.
- This step could include choosing a pattern to use, choosing a large language model (LLM), and identifying key data sources to draw from.

Second phase

1. During this stage, developers and data scientists can set their solution up to be easily monitored and iterated on. To promote observability later on, ISVs may:
 - Create golden datasets and automated multi-turn conversation datasets for copilot evaluation.
 - Debug, run, and evaluate flow with a subset of data.
 - Create variations for model evaluation with different prompts.
 - Perform pricing and resource optimization fit for LLM cost reduction to iterate and build.
2. Developers focus on performing experiments throughout development and evaluating their application's quality before deploying it to production. During this stage, it's critical to:
 - Evaluate overall application performance with test datasets using predefined metrics, including model and prompt efficacy.
 - Compare the results of these tests, establish a baseline, and then deploy the code to production.

Third phase

1. Many experiments are conducted while the application is being actively used in production. It's important to monitor your solution to ensure it's performing adequately. During this phase:
 - Telemetry instrumentation embedded within the app emits the relevant traces, metrics, and logs.
 - Cloud services emit Azure OpenAI health and other relevant metrics.

- Tailored telemetry storage holds data on traces, metrics, logs, usage, consent, and other relevant metrics.
- Pre-canned dashboard experiences enable developers, data scientists, and administrators to monitor LLM API performance and system health in the production environment.
- Feedback from end users is submitted back to developers and data scientists for evaluation to help improve the solution.

Collecting data and telemetry gives you insight into areas to address and improve in the future. By taking preemptive steps during the ideation phase, such as identifying appropriate metrics and performing rigorous evaluations of your solution while building your application, you are able to prepare your solution for success later on.

Observability challenges in generative AI

While observability in general may require ISVs to navigate many obstacles, generative AI solutions introduce specific considerations and challenges.

Qualitative evaluation

Because generative AI prompt responses are given in natural language form, they need to be evaluated uniquely. For instance, they can be checked for qualities such as groundedness and relevance.

ISVs need to consider the best route for measuring these qualities, whether they choose manual evaluation or AI-assisted metrics with a human in the loop for final validation.

No matter how you evaluate it, you also likely need preexisting datasets to compare your prompt responses to. These preparations can mean more work during development of your application as you identify an ideal response to common prompt topics.

Responsible AI

New considerations and expectations for ethical AI introduce a need to monitor privacy, security, inclusivity, and more. ISVs need to monitor these attributes in order to promote their end-user's safety, reduce risk, and minimize negative user experiences.

Microsoft's six principles of Responsible AI help promote safe, ethical, and trustworthy AI systems. To promote these values within your solution, evaluating your application against these standards is critical.

Usage and cost monitoring metrics

Tokens are the main unit of measurement for generative AI applications, and all prompts and responses are tokenized so that they can be measured. Tracking the number of tokens used is essential, as it affects the cost to run your application.

Utility metrics

Monitoring your application's user satisfaction and business impact are as critical as performance or quality metrics. Because AI interacts with customers in different ways, there are new considerations for monitoring customer engagement and retention.

Measuring the usefulness of your AI's responses can be accomplished in many different ways. For instance, prompt and response funnels track how long it takes for the interaction to result in a usable or helpful response. It's also important to consider tracking the time your user engages with the AI, the length of the conversation, and the amount of times your user accepts the provided response. In scenarios where your user can edit the response, it's essential to measure edit distance, or the extent to which they edit the response.

Performance metrics

AI requires increasingly complex, high-performance systems that must be properly maintained to ensure your solution can quickly and efficiently process prompts and data. As generative AI creates qualitative content with a large degree of variety, it's important to have systems in place to evaluate and test your AI within different scenarios.

Because LLM interactions are more complex than a typical application, they need to be measured at multiple layers to identify issues with latency. For example, the times to tokenize your user's prompt, generate a response, and return the response to a user can be measured separately, or as a whole. Each individual component of the workflow should be evaluated to identify areas for potential issues.

Your ability to observe your solution also depends on your deployment method. ISVs usually adopt one of two deployment patterns for their copilot applications. You can either deploy and manage your applications in an environment you own or deploy applications in an environment that belongs to your customers.

To learn more about how deployment affects observability across solution types, visit the [pro-code observability guide](#).

Metrics to monitor and evaluate

In the ISV realm of generative AI applications and machine learning models, it's important to continuously evaluate your solution and intervene promptly to curb undesirable behaviors. Monitoring metrics related to user experience or feedback, guardrails and Responsible AI, consistency of output, latency, and cost are essential in optimizing performance of your copilot applications.

Qualitative evaluation using AI-assisted metrics

To measure qualitative information, ISVs can use AI-assisted metrics to monitor their solutions. AI-assisted metrics utilize LLMs like GPT-4 to evaluate metrics similarly to human judgment, which provides you with more nuanced input on your solution's capabilities.

These metrics typically require parameters such as the question, answer, and any surrounding context from the conversation. They broadly fall into two categories:

1. **Risk and safety metrics** monitor for high-risk content such as violence, self-harm, sexual content, and hateful content
2. **Generation quality metrics** track qualitative measurements such as:
 - **Groundedness**-how well the model's response aligns with information from the prompt or input source.
 - **Relevance**-how well the model's response relates to the original prompt.
 - **Coherence**-the extent to which the model's response is understandable and human-like.
 - **Fluency**-the linguistics, grammar, and syntax of the model's response.

Metrics like these allow ISVs to more easily evaluate the quality of their application's responses. They provide a quick and measurable evaluation of many different values that can be difficult to interpret.

Responsible AI Standards

Microsoft is committed to upholding standards for Responsible AI. To support this, we established a set of [Responsible AI standards](#) that can help you mitigate risks associated with generative AI:

- Accountability
- Transparency
- Fairness

- Inclusivity
- Reliability and safety
- Privacy and security

ISVs can monitor metrics that notify them when an issue arises. These notifications could include qualitative AI-assisted metrics that screen responses or prompts for harmful content, or alert ISVs to certain errors or flagged messages.

For example, Azure OpenAI offers solutions that can measure the percentage of filtered prompts and responses that didn't return content due to content filtering. ISVs should monitor for prompts that return these errors and aim to reduce the amount they occur.

Customer usage and satisfaction

Some features of AI can be monitored similarly to other types of applications, such as monitoring customer retention and time spent using the application. However, there are many differences in monitoring customer satisfaction that apply specifically to AI:

- **User's reaction to the response.** This can be measured through metrics as simple as whether a user reacts to a response with a thumb up or down.
- **User's changes to the response.** In scenarios where your user can modify your AI's response to fit their needs, insight can be gained by monitoring how much your user modifies the response. For instance, a drafted email that the user changed drastically was likely not as helpful as a drafted email that the user sent as-is.
- **User's utilization of the response.** Consider monitoring whether your user takes an action through your application in response to the AI. If an AI suggests taking an action through your application, measure the rate of users who accept the suggestion.

The goal of many AI applications is to create a response that your user finds helpful. Using a prompt and response funnel is a common way to measure how quickly your solution can generate a useful response. This funnel measures the amount of time and interactions it takes for your solution to create a response the user keeps or ends the conversation with.

In this concept, the funnel begins when a user submits a prompt. As the AI generates responses that the user can interact with, the funnel narrows as the responses get closer to what the user wants. For instance, the user may edit the AI response or ask for a slightly different answer. Once the user is satisfied with the interaction, they have the specific information they were looking for and the funnel ends. Measuring how many interactions it takes for your prompt to go from broad to useful and specific is helpful for determining how effective your application is to your customer.

By observing how your users engage with your solution, you can make inferences about how helpful your application is. If your users consistently utilize your LLM's outputs without taking more action, then it's likely that the response was useful to them.

Cost monitoring

As the resources required to run a generative AI application can quickly add up, it's essential to observe them consistently.

Some areas that can impact the cost optimization of your application include:

- GPU utilization
- Storage costs and considerations
- Scaling considerations

Ensuring visibility into these metrics can help you keep costs under control, while setting up alert systems or automatic processes that are related to these metrics can also be useful for prompting immediate action.

For example, the number of prompt and completion tokens that your application uses directly affects your GPU utilization and the cost to operate your solution. Closely monitoring your token usage and setting up alerts if it crosses certain thresholds can help you stay aware of your application's behavior.

Solution availability and performance

As with all solutions, consistent monitoring of AI applications can help drive a high level of performance. One major difference between generative AI applications and others is the concept of tokenization, which needs to be considered when measuring performance.

ISVs building generative AI solutions can measure:

- The time to render the first token
- The tokens rendered per second
- The requests per second that your application can manage

While all these metrics can be measured as a group, it's also important to note that LLMs have multiple layers. For example, the time it takes for your AI to generate a response is composed of the time it takes to:

1. Receive the prompt from the user
2. Process the prompt through tokenization

3. Infer any relevant missing information
4. Generate a response
5. Compile this information into a response through detokenization
6. Send this response back to the user

Measuring at each of these steps can help you identify delays and where they're occurring, allowing you to address the problem at its source.

Other generative AI evaluation techniques

Golden Datasets

A Golden Dataset is a collection of expert answers to realistic user questions that are used to provide copilot quality assurance. These answers aren't used to train your model, but they can be compared to the answers your model gives to the same user question.

While it isn't a metric you can measure, having a high-quality standardized response you can compare your LLM's responses to helps your solution outputs. In this way, [creating your own Golden Datasets](#) ↗ for evaluating copilot performance helps accelerate your copilot evaluation process.

Multi-turn conversation simulation

Manually curating evaluation datasets can be primarily limited to single-turn conversations because of the difficulty creating natural-sounding multi-turn chats. Instead of writing scripted interactions to compare your model's answers to, ISVs can develop simulated conversations to test their copilot's multi-turn conversation abilities.

This simulation could generate dialogue by allowing your AI to interact with a simple virtual user. This user would then interact with your AI through a pre-generated script of prompts or it would generate them through AI, enabling you to create a large number of test conversations to evaluate. You could also employ human evaluators to interact with the application and generate longer conversations to review.

By evaluating your application's interactions within a longer conversation, you can evaluate how effectively it identifies your user's intent and uses context throughout the conversation. As many generative AI solutions are intended to build on multiple user interactions, it's essential to evaluate how your application handles multi-turn conversations.

Developer tools to get started

ISV developers and data scientists need to use tools and metrics to evaluate their LLM solutions. Microsoft has many options available for you to explore.

Azure AI Studio

[Azure AI Studio](#) provides observability features for model management, model benchmarks, tracing, evaluation, and fine-tuning your LLM solution.

It supports [two types of automated metrics](#) to evaluate generative AI applications: traditional ML metrics and AI-assisted metrics. You can also use the [chat playground](#) and related features easily test your model.

Prompt flow

[Prompt flow](#) is a suite of development tools designed to streamline the end-to-end development cycle of LLM-based AI applications, from prototyping and testing, to deployment and monitoring. The [prompt flow SDK](#) provides:

- [Built-in evaluators](#) that support custom code-based or prompt-based evaluators via [Prompty](#) to cater to task-specific evaluation needs.
- [Prompt tracing](#) which tracks the inputs, outputs and context of prompts, and enables developers to identify the causes and origins of model issues
- [Monitoring dashboards](#) including system (for example, token usage, latency) and custom metrics from evaluation to support pre- and post-deployment observability in [Azure AI Studio](#) and [Application Insights](#).

Other tools

[Prompty](#) is a language-agnostic asset class for creating and managing LLM prompts. It enables you to speed up the development process by providing options to design, test, and enhance your solutions.

[PyRIT \(Python Risk Identification Tool for Generative AI\)](#) is Microsoft's open automation framework for red-teaming Generative AI systems. It enables you to assess the robustness of your copilots against different harm categories.

Next steps

By designing your generative AI application with observability and monitoring in mind, you can evaluate its quality from development through production. Get started with the tools available to begin developing your application or explore options for monitoring a solution that is already in production.

Additional Resources

[How to Evaluate LLMs: A Complete Metric Framework - Microsoft Research ↗](#)

Further guidance on evaluating your LLM application

[Get started in prompt flow - Azure Machine Learning | Microsoft Learn](#)

Information on setting up and beginning to work with prompt flow

Feedback

Was this page helpful?

 Yes

 No

Tutorial: Audio/Video Calling from a Custom App into a Teams Meeting

06/12/2025

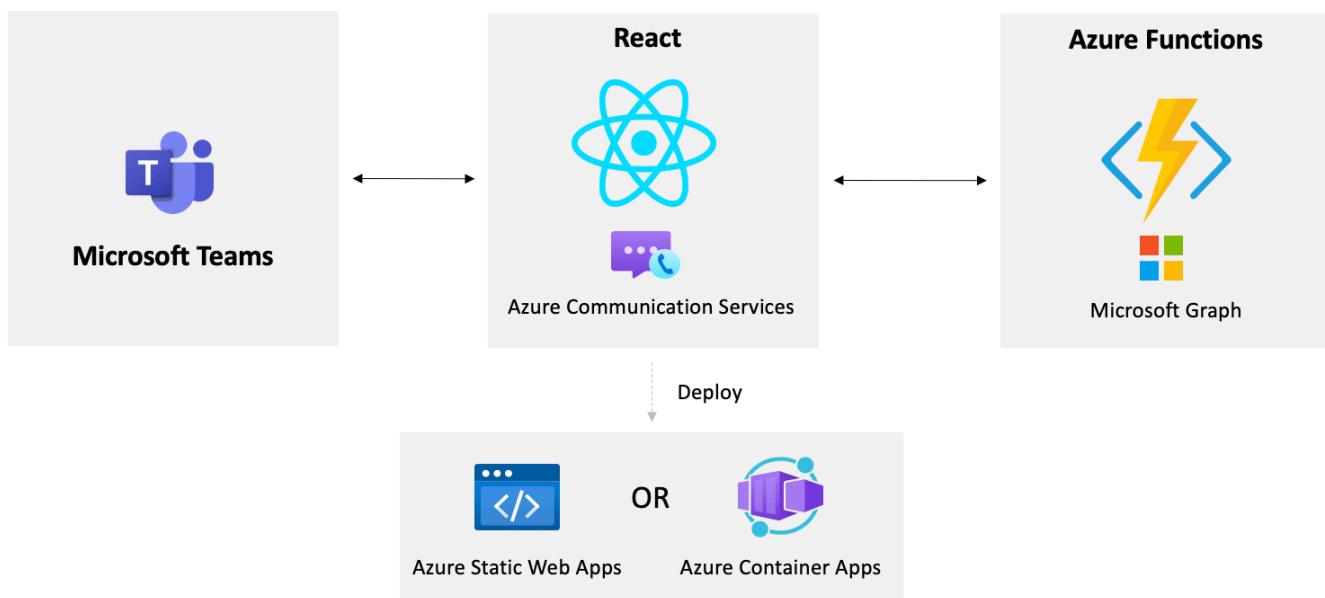
Level: Intermediate

In this tutorial, you'll learn how Azure Communication Services can be used in a custom React application to allow a user to make an audio/video call into a Microsoft Teams meeting. You'll learn about the different building blocks that can be used to make this scenario possible and be provided with hands-on steps to walk you through the different Microsoft Cloud services involved.

What You'll Build in this Tutorial

[https://www.youtube-nocookie.com/embed/5-Na9fgpVM8 ↗](https://www.youtube-nocookie.com/embed/5-Na9fgpVM8)

Overview of the Application Solution



Prerequisites

- [Node LTS ↗](#) - Node LTS is used for this project
- [git](#)
- [Visual Studio Code ↗](#) or another IDE of your choice.
- [Azure Functions Extension for VS Code ↗](#)
- [Azure Functions Core Tools](#)

- [Azure subscription ↗](#)
- [Microsoft 365 developer tenant](#)
- [GitHub account ↗](#)
- [Visual Studio ↗](#) if using the C# version of the tutorial. Visual Studio Code can also be used if preferred.

Technologies used in this tutorial include

- React
- Azure Communication Services
- Azure Functions
- Microsoft Graph
- Microsoft Teams

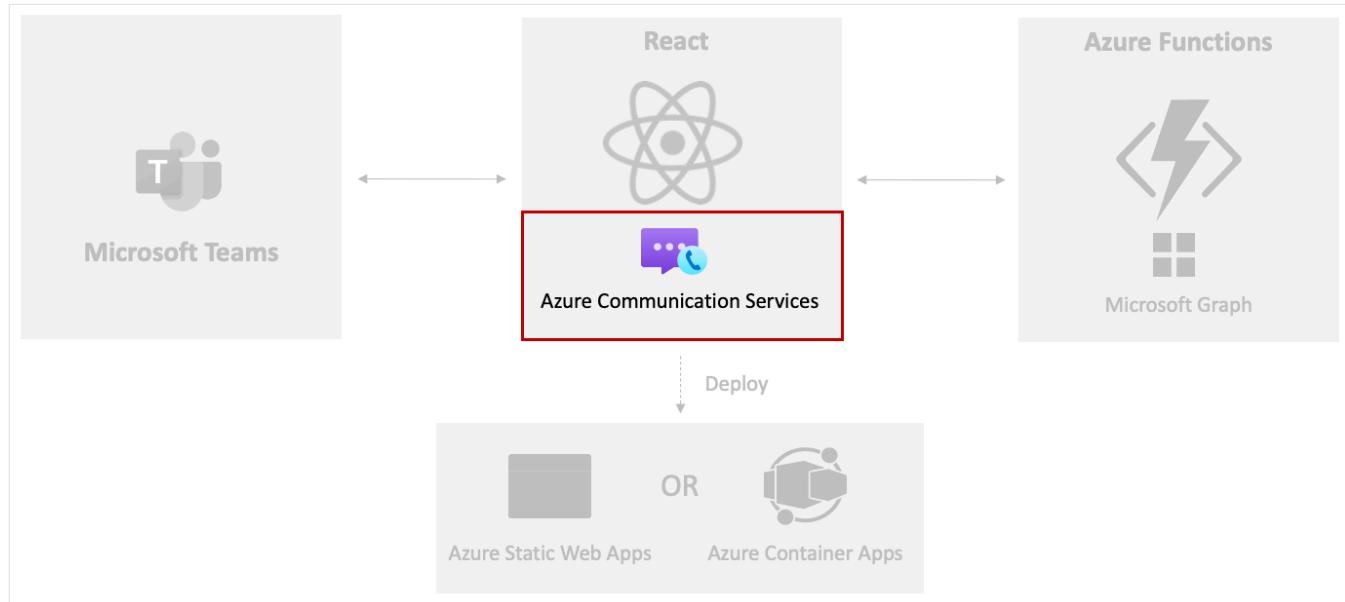
Next Step

[Create an Azure Communication Services Resource](#)

Create an Azure Communication Services Resource

06/12/2025

In this exercise you'll create an Azure Communication Services (ACS) resource in the Azure portal.



To get started, perform the following tasks:

1. Visit the [Azure portal](#) in your browser and sign in.
2. Type *communication services* in the **search bar** at the top of the page and select **Communication Services** from the options that appear.

A screenshot of the Azure search results page. The search bar at the top contains the text "communication services". Below the search bar is a navigation bar with tabs: All (selected), Services (33), Marketplace (2), Documentation (7), Resources (0), and Resource Groups (0). Under the "Services" tab, there is a list of services. The "Communication Services" option is highlighted with a red box. Other listed services include Bot Services, Cognitive Services, Free services, Kubernetes services, Email Communication Services, Analysis Services, and App Services. A "See all" link is located at the bottom right of the service list.

3. Select **Create** in the toolbar.
4. Perform the following tasks:
 - Select your Azure subscription.

- Select the resource group to use (create a new one if one doesn't exist).
- Enter an ACS resource name. It must be a unique value.
- Select a data location.

5. Select **Review + Create** followed by **Create**.

6. Once your ACS resource is created, navigate to it, and select **Settings --> Identities & User Access Tokens**.

7. Select the **Voice and video calling (VOIP)** checkbox.

8. Select **Generate**.

9. Copy the **Identity** and **User Access token** values to a local file. You'll need the values later in this exercise.



10. Select **Settings --> Keys** and copy the **Primary key** connection string value to the local file where you copied the user identity and token values.

11. To run the application you'll need a Teams meeting link. Go to [Microsoft Teams](#), sign in with your Microsoft 365 developer tenant, and select the **Calendar** option on the left.

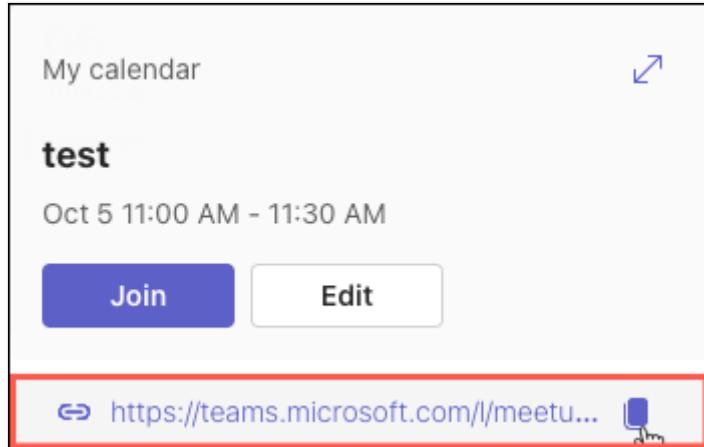
Tip

If you don't currently have a Microsoft 365 account, you can sign up for the [Microsoft 365 Developer Program](#) subscription. It's *free* for 90 days and will continually renew as long as you're using it for development activity. If you have a Visual Studio *Enterprise* or *Professional* subscription, both programs include a free Microsoft 365 [developer subscription](#), active for the life of your Visual Studio subscription.

12. Select a any date/time on the calendar, add a title for the meeting, invite a user from your Microsoft 365 developer tenant, and select **Save**.

13. Select the new meeting you added in the calendar and copy the **Teams meeting link** that is displayed into the same file where you stored the ACS user identity, token, and

connection string.



14. Now that your ACS resource is setup and you have a Teams meeting join link, let's get the React application up and running.

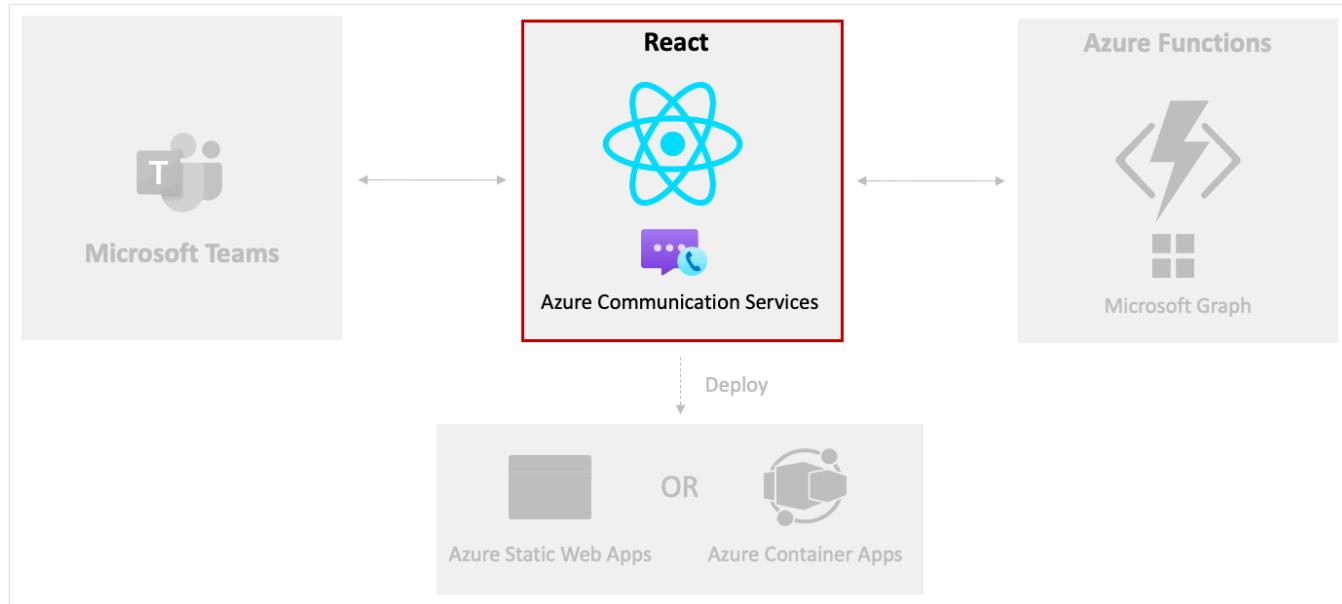
Next Step

[Integrate Azure Communication Services Calling into a React App](#)

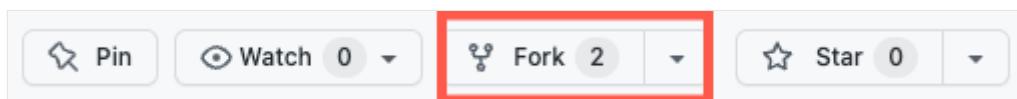
Integrate Azure Communication Services Calling into a React App

06/12/2025

In this exercise you'll add the [ACS UI calling composite](#) into a React app to enable making audio/video calls from a custom app into a Microsoft Teams meeting.



1. Visit [GitHub](#) and sign in. If you don't already have a GitHub account, you can select the **Sign up** option to create one.
2. Visit the [Microsoft Cloud GitHub Repository](#).
3. Select the **Fork** option to add the repository to your desired GitHub organization/account.



4. Run the following command to clone this repository to your machine. Replace <YOUR_ORG_NAME> with your GitHub organization/account name.

```
Console
git clone https://github.com/<YOUR_ORG_NAME>/MicrosoftCloud
```

5. Open the *samples/acs-to-teams-meeting* project folder in Visual Studio Code.
6. Expand the *client/react* folder.

7. Open the `package.json` file in VS Code and note the following ACS packages are included:

Console

```
@azure/communication-common  
@azure/communication-react
```

8. Double-check that you have `npm 10` or higher installed by opening a terminal window and running the following command:

Console

```
npm --version
```

💡 Tip

If you don't have `npm 10` or higher installed you can update npm to the latest version by running `npm install -g npm`.

9. Open a terminal window and run the `npm install` command in the `react` folder to install the application dependencies.

10. Open `App.tsx` and take a moment to explore the imports at the top of the file. These handle importing ACS security and audio/video calling symbols that will be used in the app.

TypeScript

```
import {  
    AzureCommunicationTokenCredential,  
    CommunicationUserIdentifier  
} from '@azure/communication-common';  
import {  
    CallComposite,  
    fromFlatCommunicationIdentifier,  
    useAzureCommunicationCallAdapter  
} from '@azure/communication-react';  
import React, { useState, useMemo, useEffect } from 'react';  
import './App.css';
```

ⓘ Note

You'll see how the `CallComposite` component is used later in this exercise. It provides the core UI functionality for Azure Communication Services to enable making an

audio/video call from the app into a Microsoft Teams meeting.

11. Locate the `App` component and perform the following tasks:

- Take a moment to examine the `useState` definitions in the component.
- Replace the `userId` `useState` function's empty quotes with the ACS user identity value you copied in the previous exercise.
- Replace the `token` `useState` function's empty quotes with the ACS token value you copied in the previous exercise.
- Replace the `teamsMeetingLink` `useState` function's empty quotes with the Teams meeting link value you copied in the previous exercise.

TypeScript

```
// Replace '' with the ACS user identity value
const [userId, setUserId] = useState<string>('');

// Replace '' with the ACS token value
const [token, setToken] = useState<string>('');

// Replace '' with the Teams meeting link value
const [teamsMeetingLink, setTeamsMeetingLink] = useState<string>('');
```

ⓘ Note

Later in this tutorial you'll see how to retrieve the `userId`, `token`, and `teamsMeetingLink` values dynamically.

12. Take a moment to explore the `useMemo` functions in the `App` component.

- The `credential` `useMemo` function creates a new `AzureCommunicationTokenCredential` instance once the token has a value.
- The `callAdapterArgs` `useMemo` function returns an object that has the arguments that are used to make an audio/video call. Notice that it uses the `userId`, `credential`, and `teamsMeetingLink` values in the ACS call arguments.

TypeScript

```
const credential = useMemo(() => {
  if (token) {
    return new AzureCommunicationTokenCredential(token)
  }
  return;
}, [token]);
```

```
const callAdapterArgs = useMemo(() => {
  if (userId && credential && displayName && teamsMeetingLink) {
    return {
      userId: fromFlatCommunicationIdentifier(userId) as
      CommunicationUserIdentifier,
      displayName,
      credential,
      locator: { meetingLink: teamsMeetingLink },
    }
  }
  return {};
}, [userId, credential, displayName, teamsMeetingLink]);
```

ⓘ Note

`useMemo` is used in this scenario because we only want the `AzureCommunicationTokenCredential` object and the call adapter args to be created once as the necessary parameters are passed in. View additional details about [useMemo here ↗](#).

- Once the `credentials` and `callAdapterArgs` are ready, the following line handles creating an ACS call adapter using the `useAzureCommunicationCallAdapter` React hook provided by ACS. The `callAdapter` object will be used later in the UI calling composite component.

TypeScript

```
const callAdapter = useAzureCommunicationCallAdapter(callAdapterArgs);
```

ⓘ Note

Because `useAzureCommunicationCallAdapter` is a React hook, it won't assign a value to `callAdapter` until the `callAdapterArgs` value is valid.

- Earlier you assigned the user identity, token, and Teams meeting link to state values in the `App` component. That works fine for now, but in a later exercise you'll see how to dynamically retrieve those values. Since you set the values earlier, comment out the code in the `useEffect` function as shown next. Once you get the Azure Functions running in the next exercises, you'll revisit this code.

TypeScript

```

useEffect(() => {
    /* Commenting out for now

    const init = async () => {
        setMessage('Getting ACS user');
        //Call Azure Function to get the ACS user identity and token
        let res = await fetch(process.env.REACT_APP_ACS_USER_FUNCTION as
string);
        let user = await res.json();
        setId(user.userId);
        setToken(user.token);

        setMessage('Getting Teams meeting link...');
        //Call Azure Function to get the meeting link
        res = await fetch(process.env.REACT_APP_TEAMS_MEETING_FUNCTION as
string);
        let link = await res.text();
        setTeamsMeetingLink(link);
        setMessage('');
        console.log('Teams meeting link', link);
    }
    init();

    */
}, []);

```

15. Locate the following JSX code. It uses the `CallComposite` symbol you saw imported to render the user interface used to make an audio/video call from the React app into a Teams meeting. The `callAdapter` you explored earlier is passed to its `adapter` property to provide the required arguments.

```

jsx

if (callAdapter) {
    return (
        <div>
            <h1>Contact Customer Service</h1>
            <div className="wrapper">
                <CallComposite
                    adapter={callAdapter}
                />
            </div>
        </div>
    );
}

```

16. Save the file before continuing.

17. Run `npm start` in your terminal window to run the application. Ensure you run the command within the *react* folder.
18. After the applications builds you should see a calling UI displayed. Enable selecting your microphone and camera and initiate the call. You should see that you're placed in a waiting room. If you join the meeting you setup earlier in Microsoft Teams, you can allow the guest to enter the meeting.
19. Press `ctrl + c` to stop the application. Now that you've successfully run it, let's explore how you can dynamically get the ACS user identity and token and automatically create a Microsoft Teams meeting and return the join URL using Microsoft Graph.

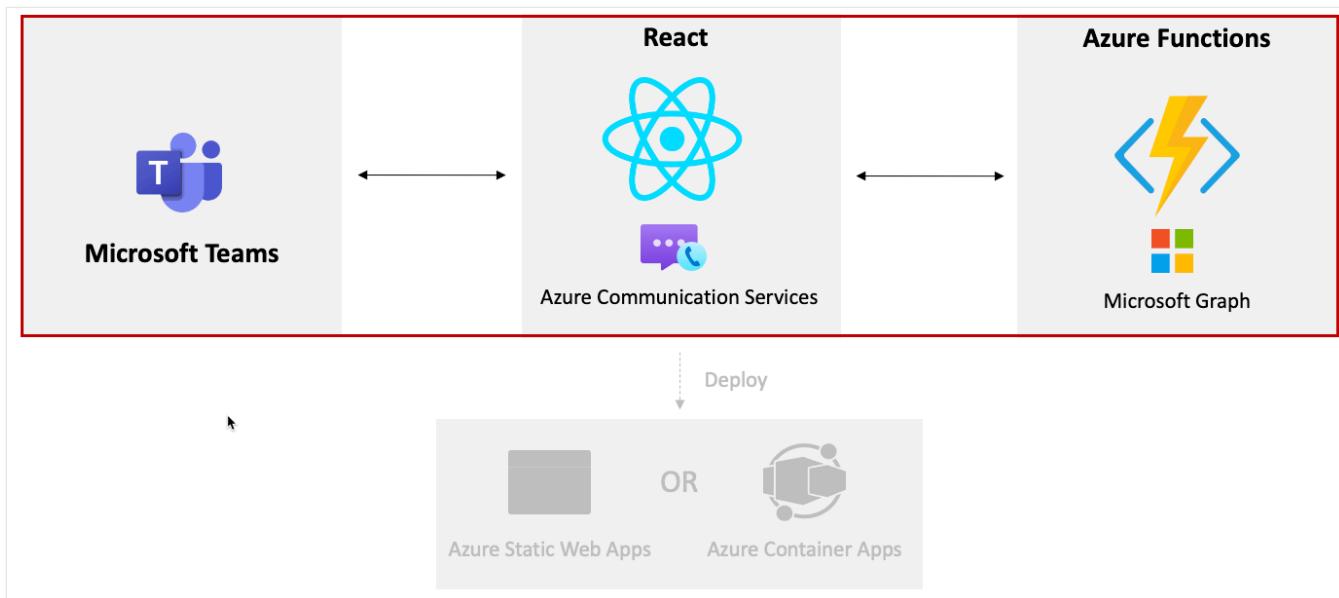
Next Step

[Dynamically Create a Microsoft Teams Meeting using Microsoft Graph](#)

Dynamically Create a Microsoft Teams Meeting using Microsoft Graph

06/12/2025

In this exercise, you'll automate the process of creating a Microsoft Teams meeting link and passing to the ACS by using Azure Functions and Microsoft Graph.



1. You'll need to create a Microsoft Entra ID app for Daemon app authentication. In this step, authentication will be handled in the background with *app credentials*, and a Microsoft Entra ID app will use Application Permissions to make Microsoft Graph API calls. Microsoft Graph will be used to dynamically create a Microsoft Teams meeting and return the Teams meeting URL.
2. Perform the following steps to create a Microsoft Entra ID app:
 - a. Go to [Azure portal](#) and select **Microsoft Entra ID**.
 - b. Select the **App registration** tab followed by **+ New registration**.
 - c. Fill in the new app registration form details as shown below and select **Register**:
 - Name: *ACS Teams Interop App*
 - Supported account types: *Accounts in any organizational directory (Any Microsoft Entra ID directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)*
 - Redirect URI: leave this blank
 - d. After the app is registered, go to **API permissions** and select **+ Add a permission**.
 - e. Select **Microsoft Graph** followed by **Application permissions**.
 - f. Select the `Calendars.ReadWrite` permission and then select **Add**.
 - g. After adding the permissions, select **Grant admin consent for** `<YOUR_ORGANIZATION_NAME>`.

- h. Go to the **Certificates & secrets** tab, select + New client secret, and then select Add.
- i. Copy the value of the secret into a local file. You'll use the value later in this exercise.
- j. Go to the **Overview** tab and copy the **Application (client) ID** and **Directory (tenant) ID** values into the same local file that you used in the previous step.

Creating a *local.settings.json* File

C#

1. Open the `samples/acs-to-teams-meeting/server/csharp/GraphACSFuctions.sln` in Visual Studio or open the *GraphACSFuctions* folder in Visual Studio Code.
2. Go to the `GraphACSFuctions` project and create a `local.settings.json` file with the following values:

JSON

```
{  
    "IsEncrypted": false,  
    "Values": {  
        "FUNCTIONS_WORKER_RUNTIME": "dotnet-isolated",  
        "TENANT_ID": "",  
        "CLIENT_ID": "",  
        "CLIENT_SECRET": "",  
        "USER_ID": "",  
        "ACS_CONNECTION_STRING": ""  
    },  
    "Host": {  
        "LocalHttpPort": 7071,  
        "CORS": "*",  
        "CORSCredentials": false  
    },  
    "watchDirectories": [  
        "Shared"  
    ]  
}
```

- Use the values you copied into the local file in an earlier exercise to update the `TENANT_ID`, `CLIENT_ID` and `CLIENT_SECRET` values.
- Define `USER_ID` with the user id that you'd like to create a Microsoft Teams Meeting.

You can get the User ID from the [Azure portal](#).

- Login using your Microsoft 365 developer tenant admin account.

- Select Microsoft Entra ID
- Navigate to the **Users** tab on the side bar.
- Search for your user name and select it to see the user details.
- Inside the user details, `Object ID` represents the `User ID`. Copy the `Object ID` value and use it for the `USER_ID` value in `local.settings.json`.

The screenshot shows the Microsoft Azure portal with the URL https://portal.azure.com/#view/Microsoft_AAD_UsersAndTenants/UserProfileMenuBlade/-/overview/userId/8fe146cd-21df-4ae1-9aca-375012b13d96. The page displays user details for Ayca Bas, including her basic info, roles, and sign-in logs. The `Object ID` field is highlighted with a red box.

ⓘ Note

`ACS_CONNECTION_STRING` will be used in the next exercise so you don't need to update it yet.

3. Open `GraphACSFuctions.sln` located in the `acs-to-teams-meeting/server/csharp` folder and note that it includes the following Microsoft Graph and Identity packages:

XML

```
<PackageReference Include="Azure.Communication.Identity" Version="1.3.1" />
<PackageReference Include="Azure.Identity" Version="1.11.2" />
<PackageReference Include="Microsoft.Graph" Version="5.51.0" />
```

4. Go to `Program.cs` and note the following code in the `ConfigureServices` method:

C#

```
var host = new HostBuilder()
    .ConfigureFunctionsWebApplication()
    .ConfigureServices(services => {
        services.AddApplicationInsightsTelemetryWorkerService();
```

```

        services.ConfigureFunctionsApplicationInsights();
        services.AddSingleton<static p =>
    {
        var config = p.GetRequiredService< IConfiguration>();
        var clientSecretCredential = new ClientSecretCredential(
            config.GetValue<string>("TENANT_ID"),
            config.GetValue<string>("CLIENT_ID"),
            config.GetValue<string>("CLIENT_SECRET")
        );

        return new GraphServiceClient(
            clientSecretCredential,
            ["https://graph.microsoft.com/.default"]
        );
    });
}

...
services.AddSingleton<IGraphService, GraphService>();
})
.Build();
}

```

- This code creates a `GraphServiceClient` object that can be used to call Microsoft Graph from Azure Functions. It's a singleton and can be injected into other classes.
- You can make Microsoft Graph API calls with `app-only permissions` (such as `Calendars.ReadWrite`) by passing a `ClientSecretCredential` value to the `GraphServiceClient` constructor. The `ClientSecretCredential` uses the `Tenant Id`, `Client Id` and `Client Secret` values from the Microsoft Entra ID app.

5. Open *Services/GraphService.cs*.

6. Take a moment to explore the `CreateMeetingEventAsync` method:

```

C#
using System;
using System.Threading.Tasks;
using Microsoft.Graph;
using Microsoft.Extensions.Configuration;

namespace GraphACSFuctions.Services;

public class GraphService : IGraphService
{
    private readonly GraphServiceClient _graphServiceClient;
    private readonly IConfiguration _configuration;

    public GraphService(GraphServiceClient graphServiceClient,

```

```

    IConfiguration configuration)
    {
        _graphServiceClient = graphServiceClient;
        _configuration = configuration;
    }

    public async Task<string> CreateMeetingAsync()
    {
        var userId = _configuration.GetValue<string>("USER_ID");
        var newMeeting = await _graphServiceClient
            .Users[userId]
            .Calendar
            .Events
            .PostAsync(new()
            {
                Subject = "Customer Service Meeting",
                Start = new()
                {
                    DateTime = DateTime.UtcNow.ToString("yyyy-MM-
ddTHH:mm:ss"),
                    TimeZone = "UTC"
                },
                End = new()
                {
                    DateTime =
                    DateTime.UtcNow.AddHours(1).ToString("yyyy-MM-ddTHH:mm:ss"),
                    TimeZone = "UTC"
                },
                IsOnlineMeeting = true
            });
        return newMeeting.OnlineMeeting.JoinUrl;
    }
}

```

- `GraphServiceClient` and `IConfiguration` objects are injected into the constructor and assigned to fields.
- The `CreateMeetingAsync()` function posts data to the [Microsoft Graph Calendar Events API](#), which dynamically creates an event in a user's calendar and returns the join URL.

7. Open `TeamsMeetingFunctions.cs` and take a moment to examine it's constructor. The `GraphServiceClient` you looked at earlier is injected and assigned to the `_graphService` field.

C#

```

private readonly IGraphService _graphService;

public TeamsMeetingFunction(IGraphService graphService) => _graphService
= graphService;

```

8. Locate the `Run` method:

```
C#  
  
[Function("HttpTriggerTeamsUrl")]
public async Task<HttpResponseData> Run(
    [HttpTrigger(AuthorizationLevel.Anonymous, "get", Route = null)]
HttpRequestData req,
    ILogger log)
{
    var response = req.CreateResponse(HttpStatusCode.OK);
    await response.WriteStringAsync(await
_graphService.CreateMeetingAsync());
    return response;
}
```

- It defines a function name of `HttpTriggerTeamsUrl` that can be called with an HTTP GET request.
- It calls `_graphService.CreateMeetingAsync()`, which creates a new event and returns the join URL.

9. Run the program by pressing `F5` in Visual Studio or by selecting **Debug --> Start Debugging** from the menu. This action starts the Azure Functions project and makes the `ACSTokenFunction` available to call.

 **Note**

If you're using VS Code you can open a terminal window in the `GraphACSFunctions` folder and run `func start`. This assumes that you have the [Azure Functions Core Tools](#) installed on your machine.

Calling the Azure Function from React

1. Now that the `httpTriggerTeamsUrl` function is ready to use, let's call it from the React app.
2. Expand the `client/react` folder.
3. Add an `.env` file into the folder with the following values:

```
REACT_APP_TEAMS_MEETING_FUNCTION=http://localhost:7071/api/httpTriggerTeamsUr  
l
```

```
REACT_APP_ACS_USER_FUNCTION=http://localhost:7071/api/httpTriggerAcsToken
```

These values will be passed into React as it builds so that you can easily change them as needed during the build process.

4. Open *client/react/App.tsx* file in VS Code.

5. Locate the `teamsMeetingLink` state variable in the component. Remove the hardcoded teams link and replace it with empty quotes:

TypeScript

```
const [teamsMeetingLink, setTeamsMeetingLink] = useState<string>('');
```

6. Locate the `useEffect` function and change it to look like the following. This handles calling the Azure Function you looked at earlier which creates a Teams meeting and returns the meeting join link:

TypeScript

```
useEffect(() => {
  const init = async () => {
    /* Commenting out for now
    setMessage('Getting ACS user');
    //Call Azure Function to get the ACS user identity and token
    const res = await fetch(process.env.REACT_APP_ACS_USER_FUNCTION as
      string);
    const user = await res.json();
    setId(user.userId);
    setToken(user.token);
    */

    setMessage('Getting Teams meeting link...');

    //Call Azure Function to get the meeting link
    const resTeams = await
      fetch(process.env.REACT_APP_TEAMS_MEETING_FUNCTION as string);
    const link = await resTeams.text();
    setTeamsMeetingLink(link);
    setMessage('');
    console.log('Teams meeting link', link);

  }
  init();
}, []);
```

7. Save the file before continuing.

8. Open a terminal window, `cd` into the *react folder, and run `npm start` to build and run the application.
9. After the application builds and loads, you should see the ACS calling UI displayed and can then call into the Teams meeting that was dynamically created by Microsoft Graph.
10. Stop the terminal process by entering `ctrl + c` in the terminal window.
11. Stop the Azure Functions project.

Next Step

[Dynamically Create an Azure Communication Services Identity and Token](#)

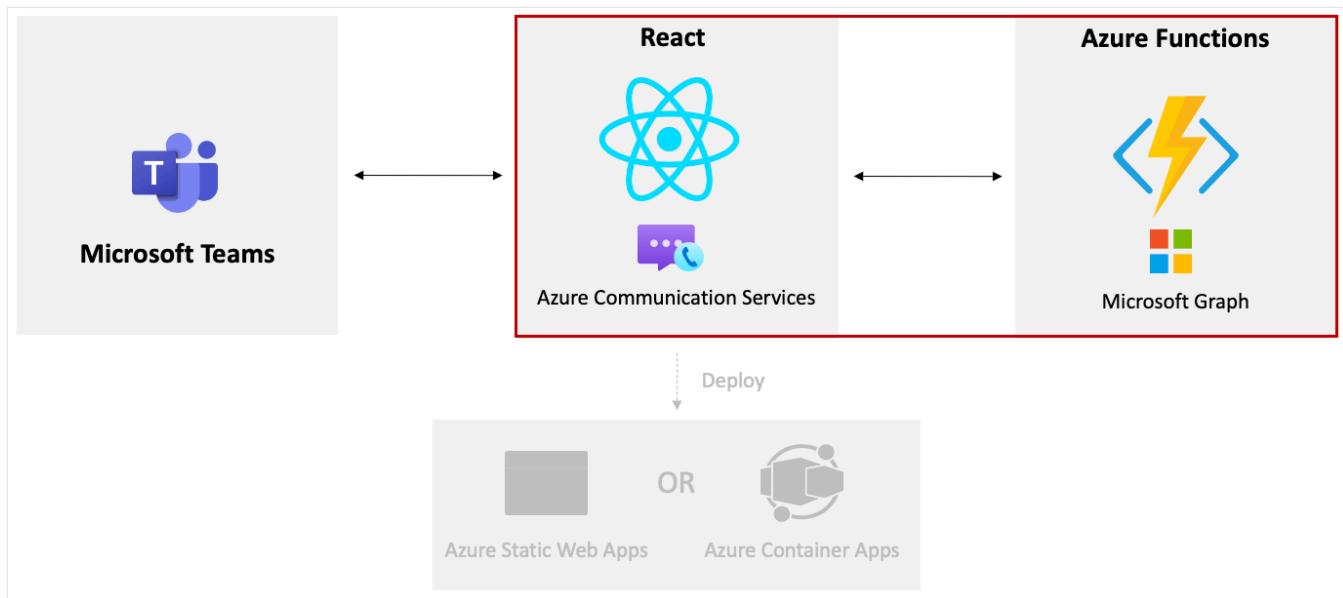
 Note

Visit the Azure Communication Services documentation to learn more about [extending Microsoft Teams meetings](#) in other ways.

Dynamically Create an Azure Communication Services Identity and Token

06/12/2025

In this exercise you'll learn how to dynamically retrieve user identity and token values from Azure Communication Services using Azure Functions. Once retrieved, the values will be passed to the ACS UI composite to enable a call to be made by a customer.



C#

1. Open `local.settings.json` and update the `ACS_CONNECTION_STRING` value with the ACS connection string you saved in an earlier exercise.
2. Open `Startup.cs` in Visual Studio and explore the second `AddSingleton()` call in the `ConfigureServices()` method.

C#

```
var host = new HostBuilder()
    .ConfigureFunctionsWebApplication()
    .ConfigureServices(services => {

    ...

    services.AddSingleton(static p =>
    {
        var config = p.GetRequiredService< IConfiguration>();
        var connectionString = config.GetValue<string>
```

```

        ("ACS_CONNECTION_STRING");
            return new CommunicationIdentityClient(connectionString);
        });

        ...
    })
    .Build();
}

```

3. The `AddSingleton()` call creates a `CommunicationIdentityClient` object using the `ACS_CONNECTION_STRING` value from `local.settings.json`.

4. Open `ACSTokenFunction.cs` and locate the constructor and field definitions.

- A field named `Scopes` is defined that includes the `CommunicationTokenScope.VoIP` scope. This scope is used to create the access token for the video call.

C#

```

private static readonly CommunicationTokenScope[] Scopes =
[
    CommunicationTokenScope.VoIP,
];

```

- The `CommunicationIdentityClient` singleton instance created in `Startup.cs` is injected into the constructor and assigned to the `_tokenClient` field.

C#

```

private readonly CommunicationIdentityClient _tokenClient;

public ACSTokenFunction(CommunicationIdentityClient tokenClient)
{
    _tokenClient = tokenClient;
}

```

5. Explore the `Run()` method in `ACSTokenFunction.cs`:

C#

```

[Function("HttpTriggerAcsToken")]
public async Task<HttpResponseData> Run(
    [HttpTrigger(AuthorizationLevel.Anonymous, "get", Route = null)]
    HttpRequestData req,
    ILogger log)
{

```

```

        var user = await _tokenClient.CreateUserAsync();
        var userToken = await _tokenClient.GetTokenAsync(user, Scopes);

        var response = req.CreateResponse(HttpStatusCode.OK);
        await response.WriteAsJsonAsync(
            new
            {
                userId = user.Value.Id,
                userToken.Value.Token,
                userToken.Value.ExpiresOn
            }
        );
        return response;
    }
}

```

- It defines a function named of `HttpTriggerAcsToken` that can be called with an HTTP GET request.
- A new ACS user is created by calling the `_tokenClient.CreateUserAsync()` method.
- An access token used for video calls is created by calling the `_tokenClient.GetTokenAsync()` method.
- The user ID and token are returned from the function as a JSON object.

6. Run the program by pressing `F5` in Visual Studio or by selecting **Debug --> Start Debugging** from the menu. This will start the Azure Functions project and make the `ACSTokenFunction` available to call.

Note

If you're using VS Code you can open a terminal window in the *GraphACSFuctions* folder and run `func start`. This assumes that you have the [Azure Functions Core Tools](#) installed on your machine.

7. Now that the Azure Functions are running locally, the client needs to be able to call into them to get the ACS user identity and token values.
8. Open *samples/acs-to-teams-meeting/client/react/App.tsx* file in your editor.
9. Locate the `userId` and `token` state variables in the component. Remove the hardcoded values and replace them with empty quotes:

TypeScript

```

const [userId, setUserId] = useState<string>('');

```

```
const [token, setToken] = useState<string>('');
```

10. Locate the `useEffect` function and change it to look like the following to enable calling the Azure Function to retrieve an ACS user identity and token:

TypeScript

```
useEffect(() => {
  const init = async () => {
    setMessage('Getting ACS user');
    //Call Azure Function to get the ACS user identity and token
    let res = await fetch(process.env.REACT_APP_ACS_USER_FUNCTION as
string);
    let user = await res.json();
    setId(user.userId);
    setToken(user.token);

    setMessage('Getting Teams meeting link...');
    //Call Azure Function to get the meeting link
    res = await fetch(process.env.REACT_APP_TEAMS_MEETING_FUNCTION as
string);
    let link = await res.text();
    setTeamsMeetingLink(link);
    setMessage('');
    console.log('Teams meeting link', link);
  }
  init();
}, []);
```

11. Save the file before continuing.

12. Open a terminal window and run `npm start` in the `react` folder. After it builds and loads you should see the ACS calling UI displayed and you can call into the Teams meeting that was dynamically created by Microsoft Graph.

13. Stop the React app by pressing `ctrl + c` in the terminal window.

14. Stop the Azure Functions project.

15. Commit your git changes and push them to your GitHub repository using Visual Studio Code:

- Select the **Source Control** icon (3rd one down in the Visual Studio Code toolbar).
- Enter a commit message and select **Commit**.
- Select **Sync Changes**.

Next Step

Deploy the App to Azure Functions and Azure Container Apps

Deploy the App to Azure Functions and Azure Container Apps

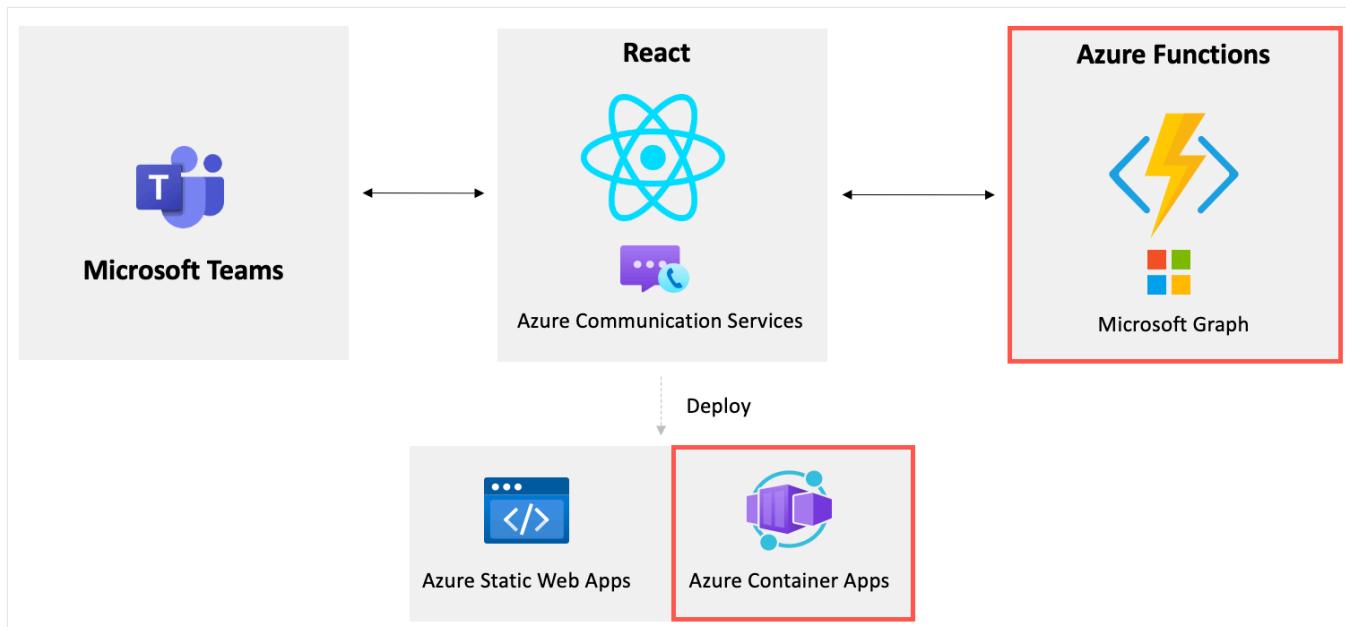
06/12/2025

ⓘ Important

In addition to the [pre-requisites listed for this tutorial](#), you'll also need to install the following tools on your machine to complete this exercise.

- [Azure CLI](#)
- If you're using VS Code, install the [Azure Functions extension](#) ↗

In this exercise you'll learn how to deploy the Microsoft Graph and ACS functions discussed in earlier exercises to Azure Functions. You'll also build a container image and deploy it to Azure Container Apps.



Deploy to Azure Functions

C#

ⓘ Note

This section uses [Visual Studio](#) to publish the C# functions to Azure. If you'd prefer to use [Visual Studio Code](#), you can follow the instructions in the [Create a C# function in Azure using Visual Studio Code](#) quickstart.

1. Open the `samples/acs-video-to-teams-meeting/server/csharp/GraphACSFuctions.sln` project in Visual Studio.
2. Right-click on the `GraphACSFuctions` project and select **Publish**.
3. Select **Azure** in the target section, then click **Next**.
4. Select **Azure Function App (Windows)**, then click **Next**.
5. Select your subscription, then select **+ Create New**.
6. Enter the following information:
 - Function name: A globally unique name is required. Example: `acsFunctions<YOUR_LAST_NAME>`.
 - Subscription: Select your subscription.
 - Resource Group: Select a resource group you created earlier in this tutorial, or you can also create a new one.
 - Hosting Plan: Consumption plan.
 - Location: Select the region you'd like to deploy to.
 - Azure Storage: Create a new one. (You can also select an existing storage account.)
 - Azure Insights: Create a new one. (You can also select an existing storage account.)

 **Note**

A globally unique name is required. You can make the name more unique by adding a number or your last name to the end of the name.

7. Once the Azure Function App is created you'll see a confirmation screen. Make sure the **Run from package** option is selected, then select **Finish**.
8. Select **Publish** to deploy the function to Azure.
9. Once the function is deployed to Azure, go to the Azure portal and select the Function App you created.
10. Copy the URL for the function you deployed. You'll use the value later in this exercise.

11. Select **Settings** --> **Configuration** in the left menu.
12. Select **+ New application setting** button and add the following keys and values in the **Application settings**. You can retrieve these values from `local.settings.json` in the `GraphACSFuctions` project.

```
# Retrieve these values from local.settings.json
TENANT_ID: <YOUR_VALUE>
CLIENT_ID: <YOUR_VALUE>
CLIENT_SECRET: <YOUR_VALUE>
USER_ID: <YOUR_VALUE>
ACS_CONNECTION_STRING: <YOUR_VALUE>
```

13. Select the **Save** button to save the settings.
14. Finally, you need to enable CORS (Cross-Origin Resource Sharing) for the function app to make the function app's APIs accessible from outside of your domain. Select **Settings** --> **CORS** in the left menu.
15. Enter `*` (accessible from any domains) in the **Allowed Origins** textbox, then select the **Save** button.

Deploy to Azure Container Apps

1. The first task you'll perform is to create a new [Azure Container Registry \(ACR\)](#) resource. Once the registry is created, you'll build an image and push it to the registry.
2. Open a command window and run the following command to login to your Azure subscription:

```
Console

az login
```

3. Add the following shell variables substituting your values for the placeholders as appropriate. Add your `<GITHUB_USERNAME>` as a lowercase value and substitute your Azure Functions domain for the `<AZURE_FUNCTIONS_DOMAIN>` value (include the `https://` in the domain value).

```
Bash
```

Bash

```
GITHUB_USERNAME=<YOUR_GITHUB_USERNAME>
RESOURCE_GROUP=<YOUR_RESOURCE_GROUP_NAME>
ACR_NAME="aca"$GITHUB_USERNAME
AZURE_FUNCTIONS_DOMAIN=<YOUR_AZURE_FUNCTIONS_URL>"
```

4. Create a new *Azure Container Registry* resource by running the following command:

Bash

Bash

```
az acr create \
--resource-group $RESOURCE_GROUP \
--name $ACR_NAME \
--sku Basic \
--admin-enabled true
```

5. Open the *client/react/Dockerfile* file in your editor and notice that the following tasks are performed:

- The React application is built and assigned to the *build* stage.
- The nginx server is configured and the output of the *build* stage is copied into the nginx server image.

6. Build the container image in Azure by running the following command from the root of the *client/react* folder. Replace *<YOUR_FUNCTIONS_DOMAIN>* with your Azure Functions domain that you copied to a local file earlier in this exercise.

Bash

```
az acr build --registry $ACR_NAME --image acs-to-teams-meeting \
--build-arg
REACT_APP_ACS_USER_FUNCTION=$AZURE_FUNCTIONS_DOMAIN/api/httpTriggerAcsTok
en \
--build-arg
REACT_APP_TEAMS_MEETING_FUNCTION=$AZURE_FUNCTIONS_DOMAIN/api/httpTriggerT
eamsUrl .
```

7. Run the following command to list the images in your registry. You should see your new image listed.

Console

```
az acr repository list --name $ACR_NAME --output table
```

8. Now that the image is deployed, you need to create an Azure Container App that can run the container.

9. Visit the [Azure portal](#) in your browser and sign in.

10. Type *container apps* in the top search bar and select **Container Apps** from the options that appear.

11. Select **Create** in the toolbar.

 **Note**

Although you're using the Azure portal, a Container App can also be created by using the Azure CLI. For more information, see [Quickstart: Deploy your first container app](#). You'll see an example of how the Azure CLI can be used at the end of this exercise as well.

12. Perform the following tasks:

- Select your subscription.
- Select the resource group to use (create a new one if needed). You can use the same resource group that you used for your ACS resource if you'd like. Copy your resource group name to the same local file where you stored your Azure Functions domain.
- Enter a Container app name of **acs-to-teams-meeting**.
- Select a region.
- Select **Create new** in the **Container Apps Environment** section.
- Enter an **Environment name** of **acs-to-teams-meeting-env**.
- Select the **Create** button.
- Select **Next: App settings >**.

13. Enter the following values in the **Create Container App** screen:

- Deselect the **Use quickstart image** checkbox.
- Name: **acs-to-teams-meeting**
- Image source: **Azure Container Registry**

- Registry: <YOUR_ACR_REGISTRY_NAME>.azurecr.io
- Image: acs-to-teams-meeting
- Image tag: latest
- CPU and Memory: 0.25 CPU cores, .5 Gi memory

14. In the **Application ingress settings** section, do the following:

- Select the **Enabled** checkbox.
- Select the **Accepting traffic from anywhere** radio button.

This will create an entry point (ingress) for your React application and allow it to be called from anywhere. Azure Container Apps redirects all traffic to HTTPS.

- Target Port: 80

15. Select **Review + create**. Once validation passes, select the **Create** button.

If you get an error it may be due to your container apps environment being inactive for too long. The simplest solution will be to go through the process of creating the container app again. Alternatively, you can run the following command to create the container app using the Azure CLI:

Bash

Bash

```
az containerapp create --name acs-to-teams-meeting --resource-group $RESOURCE_GROUP \
    --location westus --image acs-to-teams-meeting \
    --cpu 0.25 --memory 0.5 --environment-name acs-to-teams-meeting-env \
    --ingress-enabled true --ingress-target-port 80 --ingress-type External \
    --ingress-protocol Https --ingress-traffic Anywhere
```

16. Once your container app deployment completes, navigate to it in the Azure portal and select the **Application Url** on the **Overview** screen to view the application running in the nginx container!

Next Step

Congratulations!

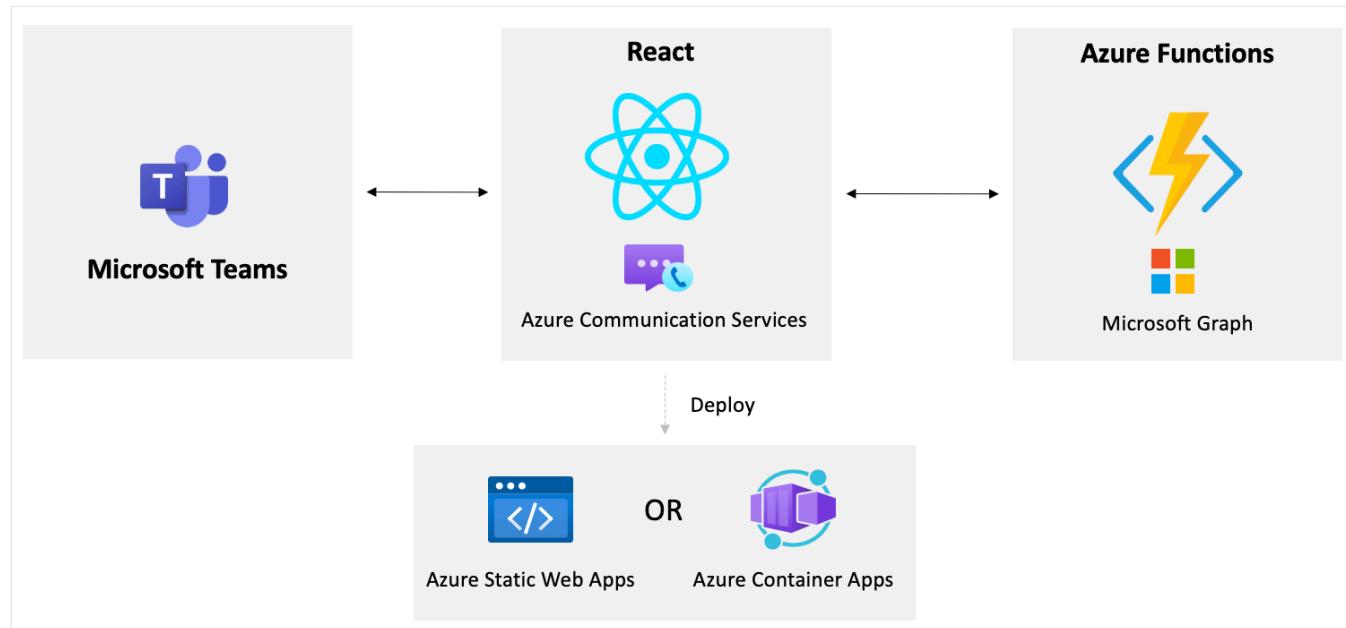
Congratulations!

06/12/2025

You've successfully integrated Azure Communication Services with Microsoft Teams meetings! In this tutorial, you learned how to:

- Create an Azure Communication Services resource.
- Add the Azure Communication Services React UI composite to a custom React application.
- Create an Azure Function to dynamically generate ACS user identities and tokens.
- Deploy the ACS and Microsoft Graph functions to Azure Functions.
- Deploy the ACS React app to Azure Container Apps.

You've completed this tutorial



Build productivity apps by using Microsoft Graph Toolkit

09/04/2025

✖ Caution

The Microsoft Graph Toolkit is deprecated. The retirement period begins September 1, 2025, with full retirement planned for August 28, 2026. Developers should migrate to using the Microsoft Graph SDKs or other supported Microsoft Graph tools for building web experiences. For more information, see the [deprecation announcement](#).

Microsoft Graph Toolkit is a collection of web components and authentication providers for connecting apps to Microsoft 365 data and intelligence. This tutorial shows you how to create a web application to monitor your calendar events, to-do tasks, and files by using Microsoft Graph Toolkit components and an MSAL2 provider.

How does the sample work?

This sample creates a web app that consumes the Microsoft Graph Toolkit MSAL2 provider to enable authentication through Microsoft Entra ID and uses UI components to render calendar events, to-do tasks, and files that will look and feel like native Microsoft experiences.

The screenshot shows the Microsoft One Productivity Hub interface. At the top, there's a header bar with a profile picture of Megan Bowen, a search bar, and various navigation icons. Below the header, the title "One Productivity Hub" is displayed. The main area is divided into three columns: "Calendar events", "To-do tasks", and "Files".

- Calendar events:**
 - 1:00 PM - 1:30 PM: Morning Coffee Talk
 - 3:00 PM - 3:30 PM: Megan/Alex Sync
 - 4:00 PM - 5:00 PM: Lunch with Adele
 - 6:00 PM - 6:30 PM: Solution architecture review
- To-do tasks:**
 - Tasks: + Add
 - Update the Learn module
 - Schedule a planning meeting with the product team
 - Solution Architecture Review
 - Prepare next week's Learn from the community blog post
- Files:**
 - Advocacy Plans FY23 (Modified 10:42 AM)
 - Azure Architecture Center Scenarios (Modified 10:42 AM)
 - Brainstorming notes (Modified 10:42 AM)
 - Events (Modified 10:42 AM)
 - Learn from the community initiative (Modified 10:43 AM)
 - Learn Together 2022 (Modified 10:42 AM)
 - M365 Workshop Docs (Modified 10:42 AM)
 - New marketing campaign assets (Modified 10:43 AM)

Prerequisites

- You need a Microsoft 365 developer tenant. You might qualify for one through the [Microsoft 365 Developer Program](#); for details, see the [FAQ](#). Alternatively, you can [sign up for a 1-month free trial](#) or [purchase a Microsoft 365 plan](#).
- Install [Visual Studio Code](#).
- Install [Visual Studio Code Live Server](#) to test your web app.

Register the application in Microsoft Entra ID

Register the application in the Microsoft Entra ID to enable user authentication.

1. Go to the [Microsoft Entra admin center](#) and sign in with your Microsoft 365 Developer Program tenant.
2. Expand the **Identity** menu > expand **Applications** > select **App registrations** > **New registration**.

3. Complete the **Register an application** form using the following values, and then select **Register**.

- **Name:** One Productivity Hub Demo
- **Supported account types:** Accounts in any organizational directory (Any Microsoft Entra directory - Multitenant) and personal Microsoft accounts (for example, Skype, Xbox)
- **Redirect URI:** Select **Single-page application (SPA)** as a type of redirect URI and put `http://localhost:3000/index.html` as a redirect URI to test locally.

Register an application

* Name

The user-facing display name for this application (this can be changed later).



Supported account types

Who can use this application or access this API?

- Accounts in this organizational directory only (m365advocates only - Single tenant)
 Accounts in any organizational directory (Any Azure AD directory - Multitenant)
 Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
 Personal Microsoft accounts only

[Help me choose...](#)

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.



[By proceeding, you agree to the Microsoft Platform Policies](#)

Register

4. When app registration is completed, go to the **Overview** tab in your application page and copy **Application (client) ID**. You need this ID for the following step.

Create the application

In this step, you'll create a web app and enable authentication with the Microsoft Graph Toolkit MSAL2 provider.

Create a web app

1. Create a new folder and name it **OneProductivityHub**. Right-click and open the folder with Visual Studio Code.
2. Create a new file in the **OneProductivityHub** folder and name it **index.html**.
3. Select **CTRL + SPACE** and choose **HTML sample** from the options.
4. To enable authentication with Microsoft Graph Toolkit via a CDN, add the following script tag in **index.html** inside the `<body></body>` section:

HTML

```
<script type="module">
  import { registerMgtComponents, registerMgtMsal2Provider } from
  "https://unpkg.com/@microsoft/mgt@4";
  registerMgtMsal2Provider();
  registerMgtComponents();
</script>
```

Initialize the MSAL2 provider

1. In **index.html**, add the MSAL2 provider in the `<body></body>` section as follows:

HTML

```
<mgt-msal2-provider
  client-id=<YOUR_CLIENT_ID>
  scopes="User.Read,
  User.ReadBasic.All,
  Calendars.Read,
  Files.Read,
  Files.Read.All,
  Sites.Read.All,
  Tasks.Read,
  Tasks.ReadWrite,
```

```
    People.Read">
</mgt-msal2-provider>
```

ⓘ Important

The following scopes defined in the provider will be shown as a list of required permissions to request user's consent during the authentication process: User.Read, User.ReadBasic.All, Calendars.Read, Files.Read, Files.Read.All, Sites.Read.All, Tasks.Read, Tasks.ReadWrite, People.Read.

2. Replace <YOUR_CLIENT_ID> with the client ID you copied from the Microsoft Entra application.

3. Make sure that the final version of **index.html** is similar to the following example:

HTML

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <meta http-equiv='X-UA-Compatible' content='IE=edge'>
    <title>One Productivity Hub</title>
    <meta name='viewport' content='width=device-width, initial-scale=1'>
    <link rel='stylesheet' type='text/css' media='screen' href='main.css'>
    <script src='main.js'></script>
</head>
<body>
    <script type="module">
        import { registerMgtComponents, registerMgtMsal2Provider } from
        "https://unpkg.com/@microsoft/mgt@4";
        registerMgtMsal2Provider();
        registerMgtComponents();
    </script>
    <mgt-msal2-provider
        client-id=<YOUR_CLIENT_ID>
        scopes="User.Read,
        User.ReadBasic.All,
        Calendars.Read,
        Files.Read,
        Files.Read.All,
        Sites.Read.All,
        Tasks.Read
        Tasks.ReadWrite,
        People.Read">
    </mgt-msal2-provider>
</body>
</html>
```

Design the application

In this step, you'll design your web app by using Microsoft Graph Toolkit components and style it with CSS.

Initialize the Login component

In `index.html` in the `<body></body>` section, add the following code under the provider.

HTML

```
<div>
  <mgt-login />
</div>
```

Create a title and column for the rest of the components

To make your app look structured, create a title and a column for each feature that will be added in the One Productivity Hub. In `index.html` under `<body></body>`, add the following HTML code inside the div, under the login component.

HTML

```
<div class="features">

  <div class="header"><div class="title">
    <h2>One Productivity Hub</h2>
    <div class="row">
      <div class="column"><h3>Calendar events</h3></div>
      <div class="column"><h3>To-do tasks</h3></div>
      <div class="column"><h3>Files</h3></div>
    </div>
  </div></div>

  <div class="row" id="content">
    <div class="column" id="mgt-col"></div>
    <div class="column" id="mgt-col"></div>
    <div class="column" id="mgt-col"></div>
  </div>

</div>
```

Agenda component

Under the div tagged with `class="row"`, add the Agenda component inside the first column div.

HTML

```
<mgt-agenda />
```

To-do component

Under the div tagged with `class="row"`, add the To-do component inside the second column div.

HTML

```
<mgt-todo />
```

FileList component

Under the div tagged with `class="row"`, add the File list component inside the third column div.

HTML

```
<mgt-file-list />
```

Style your web app with CSS

1. Create an `index.css` file under your project and add the following CSS code.

css

```
body,  
#root>div {  
    background-color: #F3F2F1;  
}  
.features {  
    min-height: 80vh;  
    margin: 20px;  
    background-color: #FFF;  
    box-shadow: 0px 1.2px 3.6px rgba(0, 0, 0, 0.11), 0px 6.4px 14.4px rgba(0,  
0, 0, 0.13);  
    border-radius: 4px;  
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  
}
```

```
.header {
  display: flex;
  background-color: #f0f0f0;
}
.title {
  margin-top: 20px;
  margin-left: 10px;
  width: 100%;
}
.title h2 {
  font-size: 24px;
  padding-left: 5px;
  display: inline;
  font-weight: 600;
}
.title h3 {
  float: left;
  width: 32%;
  background: transparent;
  font-size: 16px;
  margin-bottom: 10px;
  padding-left: 10px;
  padding-top: 10px;
  color: #8A8886;
  font-weight: 600;
}
mgt-login {
  margin-left: 20px;
  --avatar-size: 60px;
  --font-family: 'Segoe UI';
  --font-size: 20px;
  --font-weight: 700;
  --color: black;
  --text-transform: none;
  --line2-font-size: 14px;
  --line2-font-weight: 400;
  --line2-color: #8A8886;
  --line2-text-transform: none;
}
#content, html, body {
  height: 98%;
}
#mgt-col {
  float: left;
  width: 32%;
  background: transparent;
  height: 500px;
  overflow: hidden;
  padding: 5px;
  margin-top: 5px;
}
#mgt-col:hover {
  overflow-y: auto;
}
```

2. In **index.html** within `<head></head>`, define the stylesheet link `href` as **index.css**.

HTML

```
<link rel='stylesheet' type='text/css' media='screen' href='index.css'>
```

3. Make sure that the final version of **index.html** is similar to the following.

HTML

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <meta http-equiv='X-UA-Compatible' content='IE=edge'>
    <title>One Productivity Hub</title>
    <meta name='viewport' content='width=device-width, initial-scale=1'>
    <link rel='stylesheet' type='text/css' media='screen' href='index.css'>
    <script src='main.js'></script>
</head>
<body>
    <script type="module">
        import { registerMgtComponents, registerMgtMsal2Provider } from
"https://unpkg.com/@microsoft/mgt@4";
        registerMgtMsal2Provider();
        registerMgtComponents();
    </script>
    <mgt-msal2-provider
        client-id=<YOUR_CLIENT_ID>
        scopes="User.Read,
        User.ReadBasic.All,
        Calendars.Read,
        Files.Read,
        Files.Read.All,
        Sites.Read.All,
        Tasks.Read,
        Tasks.ReadWrite,
        People.Read">
    </mgt-msal2-provider>
    <div>
        <mgt-login />
    </div>
    <div class="features">
        <div class="header">
            <div class="title">
                <h2>One Productivity Hub</h2>
                <div class="row">
                    <div class="column"><h3>Calendar events</h3></div>
                    <div class="column"><h3>To-do tasks</h3></div>
                    <div class="column"><h3>Files</h3></div>
                </div>
            </div>
        </div>
    
```

```
</div>
<div class="row" id="content">
    <div class="column" id="mgt-col"><mgt-agenda /></div>
    <div class="column" id="mgt-col"><mgt-todo /></div>
    <div class="column" id="mgt-col"><mgt-file-list /></div>
</div>
</div>
</body>
</html>
```

Run the application

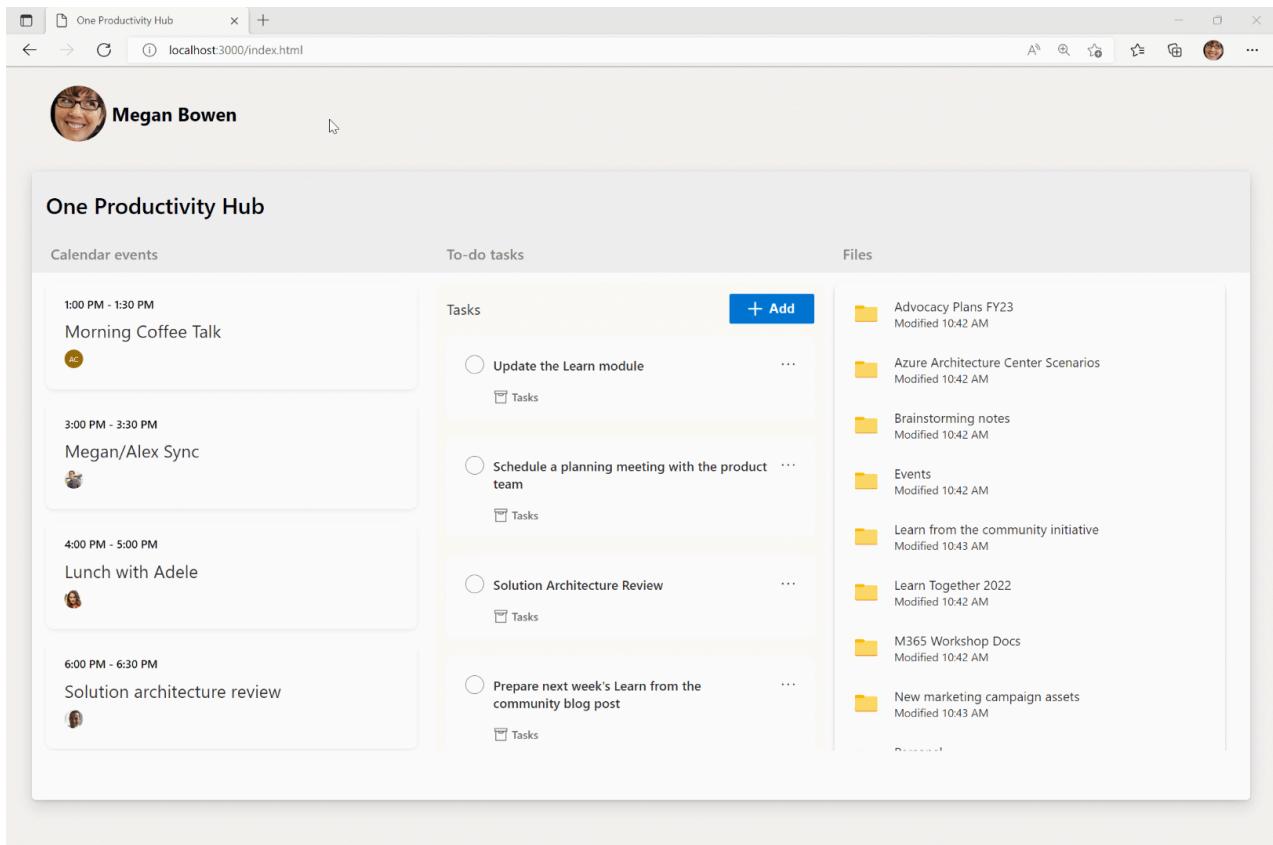
In this step, you will run your sample app in the browser by using Live Server.

1. Select **CTRL + SHIFT + P** to open the panel in Visual Studio Code, type **Preferences** on the panel and select **Preferences: Open Workspace Settings (JSON)** from the options.
2. In the **settings.json** file that opens, add the following code.

JSON

```
{
"liveServer.settings.host": "localhost",
"liveServer.settings.port": 3000
}
```

3. Select **CTRL + SHIFT + P** to open the panel in Visual Studio Code, type **Live Server** on the panel and select **Live Server: Open with Live Server** from the options. Live Server will run the app in your browser.
4. Select **Sign in** and use your Microsoft 365 Developer Program tenant to sign in.
5. Consent to the required permissions to use the application features such as viewing calendar events, to-do tasks, and file folders.
6. To make sure that One Productivity Hub app works properly, add some calendar events, to-do tasks, and file folders in your Microsoft 365 Developer tenant.



Related content

You have successfully built your One Productivity Hub sample app using Microsoft Graph Toolkit. In this tutorial, you created a web app and consumed the Microsoft Graph Toolkit MSAL2 provider and UI components.

- [Microsoft Graph Toolkit overview](#)
- [Develop apps with the Microsoft Graph Toolkit](#)

Integrate OpenAI, Communication, and Organizational Data Features into a Line of Business App

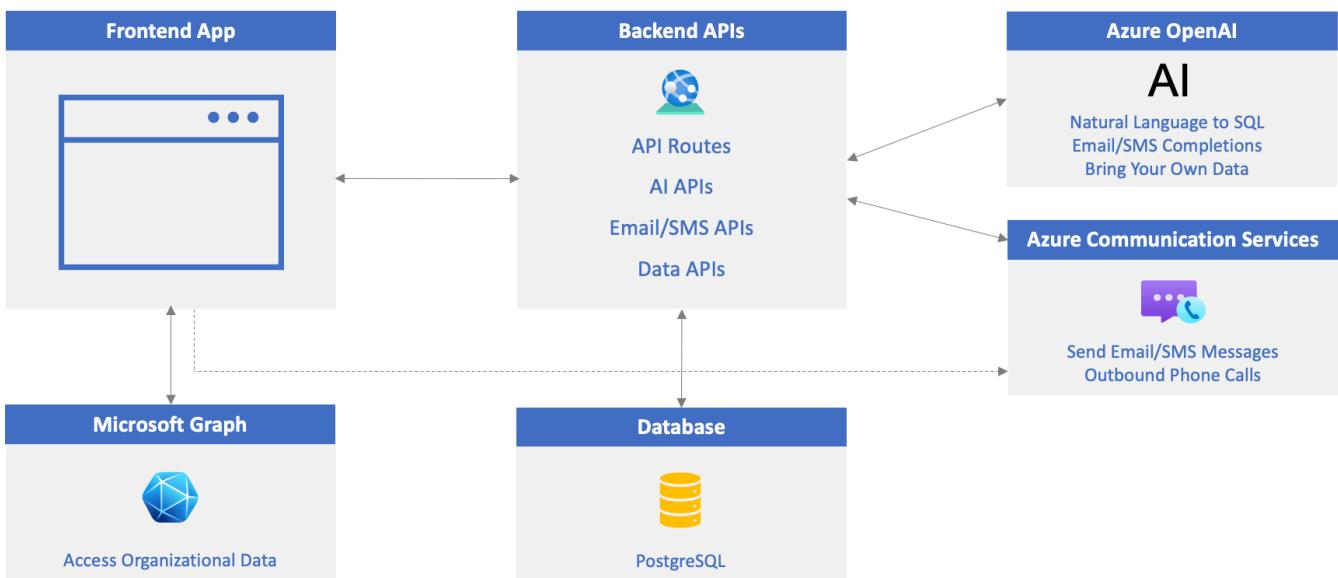
06/13/2025

Level: Intermediate

This tutorial demonstrates how Azure OpenAI, Azure Communication Services, and Microsoft Graph/Microsoft Graph Toolkit can be integrated into a Line of Business (LOB) application to enhance user productivity, elevate the user experience, and take LOB apps to the next level. Key features in the application include:

- **AI:** Enable users to ask questions in natural language and convert their answers to SQL that can be used to query a database, allow users to define rules that can be used to automatically generate email and SMS messages, and learn how natural language can be used to retrieve data from your own custom data sources. Azure OpenAI is used for these features.
- **Communication:** Enable in-app phone calling to customers and Email/SMS functionality using Azure Communication Services.
- **Organizational Data:** Pull in related organizational data that users may need (documents, chats, emails, calendar events) as they work with customers to avoid context switching. Providing access to this type of organizational data reduces the need for the user to switch to Outlook, Teams, OneDrive, other custom apps, their phone, etc. since the specific data and functionality they need is provided directly in the app. Microsoft Graph and Microsoft Graph Toolkit are used for this feature.

The application is a simple customer management app that allows users to manage their customers and related data. It consists of a front-end built using TypeScript that calls back-end APIs to retrieve data, interact with AI functionality, send email/SMS messages, and pull in organizational data. Here's an overview of the application solution that you'll walk through in this tutorial:



The tutorial will walk you through the process of setting up the required Azure and Microsoft 365 resources. It'll also walk you through the code that is used to implement the AI, communication, and organizational data features. While you won't be required to copy and paste code, some of the exercises will have you modify code to try out different scenarios.

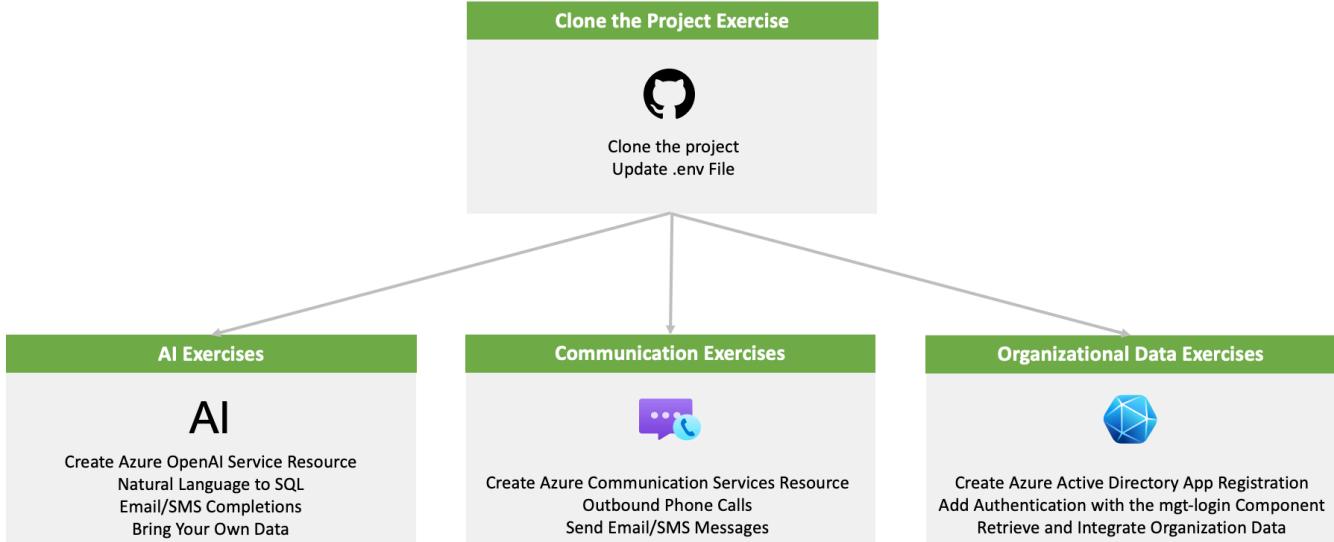
What You'll Build in this Tutorial

[https://www.youtube-nocookie.com/embed/qe3oAQEhGuA ↗](https://www.youtube-nocookie.com/embed/qe3oAQEhGuA)

Choose Your Own Adventure

You can complete the entire tutorial from start to finish or complete specific topics of interest. The tutorial is broken down into the following topics:

- [Clone the Project Exercise](#) (required exercise).
- **AI Exercises:** Create an [Azure OpenAI resource](#) and use it to convert natural language to SQL, generate email/SMS messages, and work with your own data and documents.
- **Communication Exercises:** Create an [Azure Communication Services resource](#) and use it to make phone calls from the app and send email/SMS messages.
- **Organizational Data Exercises:** [Create a Microsoft Entra ID app registration](#) so that Microsoft Graph and Microsoft Graph Toolkit can be used to authenticate and pull organizational data into the application.



Prerequisites

- [Node](#) - Node 20+ and npm 10+ will be used for this project
- [git](#)
- [Visual Studio Code](#) (although Visual Studio Code is recommended, any editor can be used)
- [Azure subscription](#)
- [Microsoft 365 developer tenant](#)
- [Docker Desktop](#) or another OCI (Open Container Initiative) compliant container runtime such as [Podman](#), or [nerdctl](#) capable of running a container.

Microsoft Cloud Technologies used in this Tutorial

- Azure Communication Services
- Azure OpenAI Service
- Microsoft Entra ID
- Microsoft Graph
- Microsoft Graph Toolkit

Next Step

[Clone the Project](#)

Clone the Project

06/13/2025

The code project used in this tutorial is available at

<https://github.com/microsoft/MicrosoftCloud>. The project's repository includes both client-side and server-side code required to run the project, enabling you to explore the integrated features related to artificial intelligence (AI), communication, and organizational data. Additionally, the project serves as resource to guide you in incorporating similar features into your own applications.

In this exercise you will:

- Clone the GitHub repository.
- Add an `.env` file into the project and update it.

Before proceeding, ensure that you have all of the prerequisites installed and configured as outlined in the [Prerequisites](#) section of this tutorial.

Clone the GitHub Repository and Create an `.env` File

1. Run the following command to clone the [Microsoft Cloud GitHub Repository](#) to your machine.

Console

```
git clone https://github.com/microsoft/MicrosoftCloud
```

2. Open the `MicrosoftCloud/samples/openai-acs-msgraph` folder in Visual Studio Code.

 **Note**

Although we'll use Visual Studio Code throughout this tutorial, any code editor can be used to work with the sample project.

3. Notice the following folders and files:

- `client`: Client-side application code.
- `server`: Server-side API code.
- `docker-compose.yml`: Used to run a local PostgreSQL database.

4. Rename the `.env.example` in the root of the project to `.env`.

5. Open the `.env` file and take a moment to look through the keys that are included:

```
ENTRAID_CLIENT_ID=
TEAM_ID=
CHANNEL_ID=
OPENAI_API_KEY=
OPENAI_ENDPOINT=
OPENAI_MODEL=gpt-4o
OPENAI_API_VERSION=2024-05-01-preview
POSTGRES_USER=
POSTGRES_PASSWORD=
ACS_CONNECTION_STRING=
ACS_PHONE_NUMBER=
ACS_EMAIL_ADDRESS=
CUSTOMER_EMAIL_ADDRESS=
CUSTOMER_PHONE_NUMBER=
API_PORT=3000
AZURE_AI_SEARCH_ENDPOINT=
AZURE_AI_SEARCH_KEY=
AZURE_AI_SEARCH_INDEX=
```

6. Update the following values in `.env`. These values will be used by the API server to connect to the local PostgreSQL database.

```
POSTGRES_USER=web
POSTGRES_PASSWORD=web-password
```

7. Now that you have the project in place, let's try out some of the application features and learn how they're built. Select the **Next** button below to continue or jump to a specific exercise using the table of contents.

Next Step

[AI: Create an Azure OpenAI Resource and Deploy a Model](#)

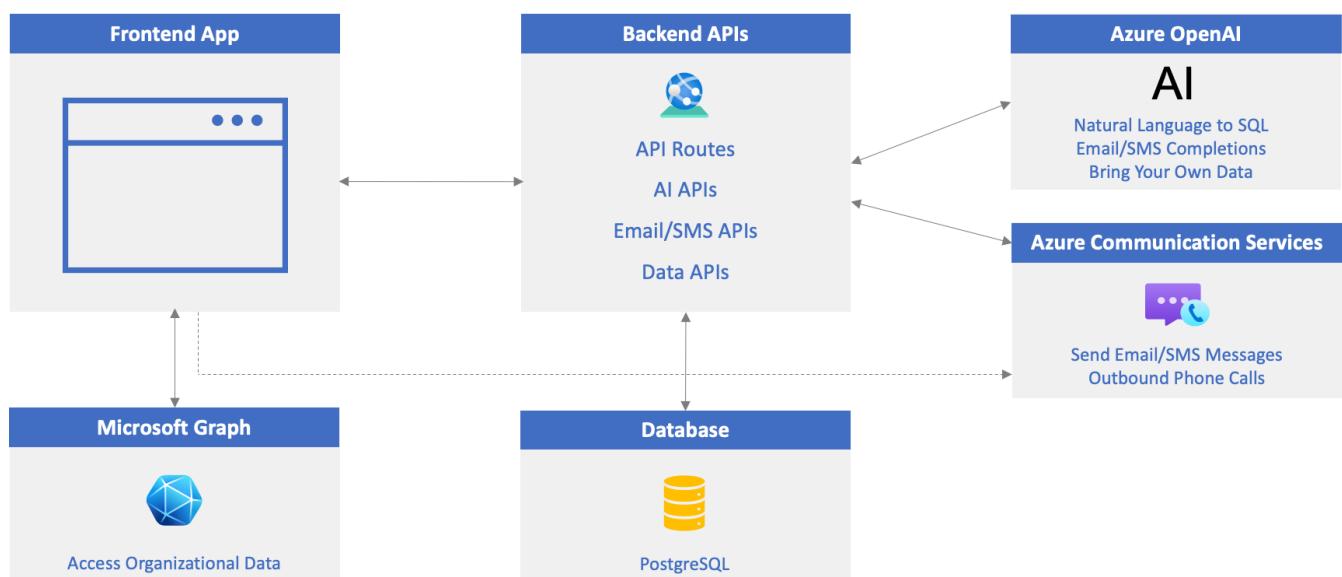
AI: Create an Azure OpenAI Resource and Deploy a Model

06/13/2025

To get started using Azure OpenAI in your applications, you need to create an Azure OpenAI Service and deploy a model that can be used to perform tasks such as converting natural language to SQL, generating email/SMS message content, and more.

In this exercise you will:

- Create an Azure OpenAI Service resource.
- Deploy a model.
- Update the `.env` file with values from your Azure OpenAI service resource.



Create an Azure OpenAI Service Resource

1. Visit the [Azure portal](#) in your browser and sign in.
2. Enter *openai* in the **search bar** at the top of the portal page and select **Azure OpenAI** from the options that appear.

The screenshot shows the Azure portal search results for the term "openai". The search bar at the top contains "openai". Below the search bar, there are several filter buttons: All (selected), Services (1), Marketplace (4), Documentation (99+), Resources (0), and Resource Groups (0). Under the "Services" heading, there is a list of items, with "Azure OpenAI" being the first item and highlighted with a red box. Other items listed include "Azure Active Directory (0)" and "Services".

3. Select **Create** in the toolbar.

ⓘ Note

While this tutorial focuses on Azure OpenAI, if you have an OpenAI API key and would like to use it, you can skip this section and go directly to the [Update the Project's .env File](#) section below. Assign your OpenAI API key to `OPENAI_API_KEY` in the `.env` file (you can ignore any other `.env` instructions related to OpenAI).

4. Azure OpenAI models are available in specific regions. Visit the [Azure OpenAI model availability](#) document to learn which regions support the gpt-4o model used in this tutorial.

5. Perform the following tasks:

- Select your Azure subscription.
- Select the resource group to use (create a new one if needed).
- Select a region where the gpt-4o model is supported based on the document you looked at earlier.
- Enter the resource name. It must be a unique value.
- Select the **Standard S0** pricing tier.

6. Select **Next** until you get to the **Review + submit** screen. Select **Create**.

7. Once your Azure OpenAI resource is created, navigate to it and select **Resource Management --> Keys and Endpoint**.

8. Locate the **KEY 1** and **Endpoint** values. You'll use both values in the next section so copy them to a local file.

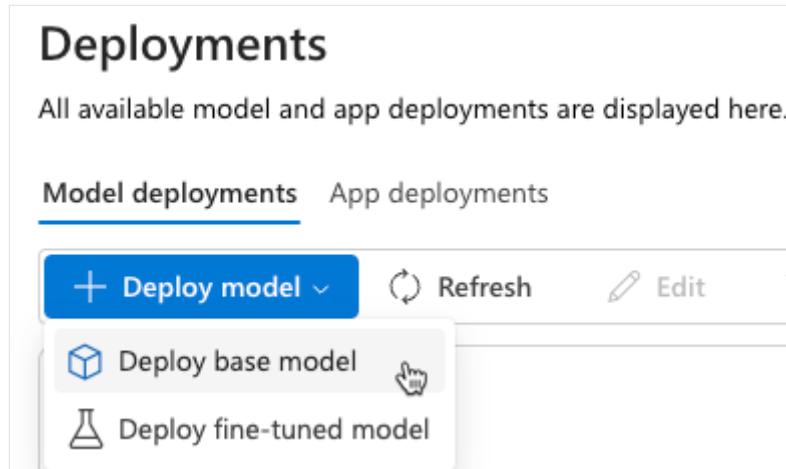
Show Keys

KEY 1	
KEY 2	
Location/Region	southcentralus	
Endpoint	https://	

9. Select **Resource Management --> Model deployments**.

10. Select the **Manage Deployments** button to go to Azure OpenAI Studio.

11. Select **Deploy model --> Deploy base model** in the toolbar.



12. Select **gpt-4o** from the list of models and select **Confirm**.

ⓘ Note

Azure OpenAI supports several different types of models. Each model can be used to handle different scenarios.

13. The following dialog will display. Take a moment to examine the default values that are provided.

Deploy model gpt-4o

Deployment name * 

Model version

Deployment type

Pay per API call with higher rate limits. Data might be processed outside of the resource's Azure geography, but data storage remains in its Azure geography.
[Learn more about data residency](#) 

Current Azure OpenAI resource
danwahlin-ai-test | westus

 450K tokens per minute quota available for your deployment

Tokens per Minute Rate Limit (thousands) 

1K

Corresponding requests per minute (RPM) = 6

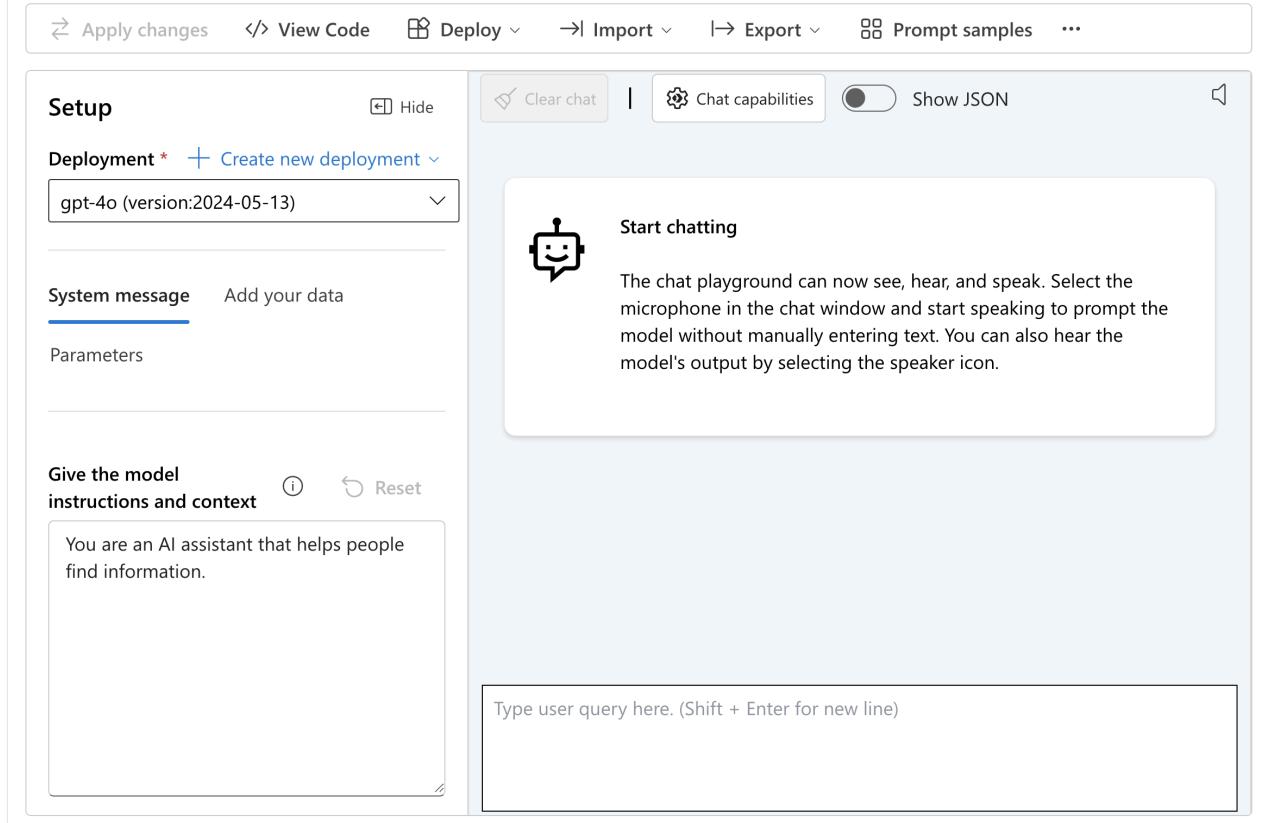
Content filter 

Deploy **Cancel**

14. Change the **Tokens per Minute Rate Limit (thousands)** value to 100K. This will allow you to make more requests to the model and avoid hitting the rate limit as you perform the steps that follow.
15. Select **Deploy**.
16. Once the model is deployed, select **Playgrounds --> Chat**.
17. The **Deployment** dropdown should display the **gpt-4o** model.

Chat playground



18. Take a moment to read through the **System message** text that's provided. This tells the model how to act as the user interacts with it.
19. Locate the textbox in the chat area and enter **Summarize what Generative AI is and how it can be used**. Select `Enter` to send the message to the model and have it generate a response.
20. Experiment with other prompts and responses. For example, enter **Provide a short history about the capital of France** and notice the response that's generated.

Update the Project's `.env` File

1. Go back to Visual Studio Code and open the `.env` file at the root of the project.
2. Copy the **KEY 1** value from your Azure OpenAI resource and assign it to `OPENAI_API_KEY` in the `.env` file located in the root of the `openai-acs-msgraph` folder:

```
OPENAI_API_KEY=<KEY_1_VALUE>
```

3. Copy the **Endpoint* value and assign it to `OPENAI_ENDPOINT` in the `.env` file. Remove the `/` character from the end of the value if it's present.

```
OPENAI_ENDPOINT=<ENDPOINT_VALUE>
```

ⓘ Note

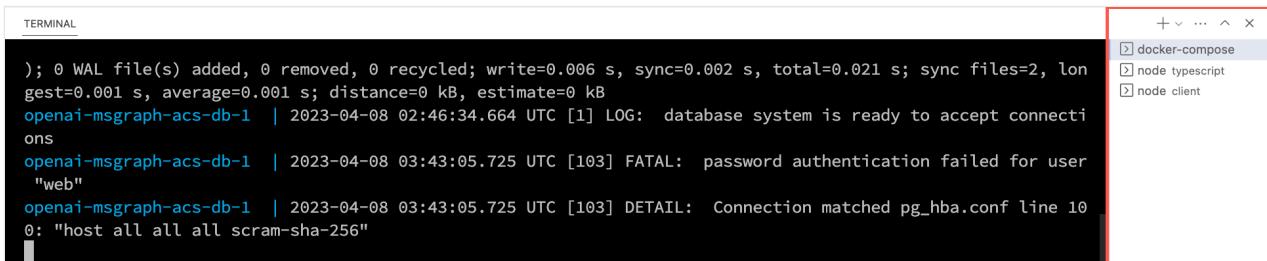
You'll see that values for `OPENAI_MODEL` and `OPENAI_API_VERSION` are already set in the `.env` file. The model value is set to `gpt-4o` which matches the model deployment name you created earlier in this exercise. The API version is set to a supported value defined in the [Azure OpenAI reference documentation](#).

4. Save the `.env` file.

Start the Application Services

It's time to start up your application services including the database, API server, and web server.

1. In the following steps you'll create three terminal windows in Visual Studio Code.



The screenshot shows a terminal window with the following log output:

```
; 0 WAL file(s) added, 0 removed, 0 recycled; write=0.006 s, sync=0.002 s, total=0.021 s; sync files=2, longest=0.001 s, average=0.001 s; distance=0 kB, estimate=0 kB
openai-msgraph-acs-db-1 | 2023-04-08 02:46:34.664 UTC [1] LOG:  database system is ready to accept connections
openai-msgraph-acs-db-1 | 2023-04-08 03:43:05.725 UTC [103] FATAL:  password authentication failed for user "web"
openai-msgraph-acs-db-1 | 2023-04-08 03:43:05.725 UTC [103] DETAIL:  Connection matched pg_hba.conf line 10
0: "host all all all scram-sha-256"
```

To the right of the terminal, a command palette is open with the following options:

- docker-compose
- node typescript
- node client

2. Right-click on the `.env` file in the Visual Studio Code file list and select **Open in Integrated Terminal**. Ensure that your terminal is at the root of the project - `openai-acs-msgraph` - before continuing.
3. Choose from **one** of the following options to start the PostgreSQL database:

- If you have [Docker Desktop](#) installed and running, run `docker-compose up` in the terminal window and press `Enter`.
- If you have Podman with [podman-compose](#) installed and running, run `podman-compose up` in the terminal window and press `Enter`.
- To run the PostgreSQL container directly using either Docker Desktop, Podman, nerdctl, or another container runtime you have installed, run the following command in the terminal window:

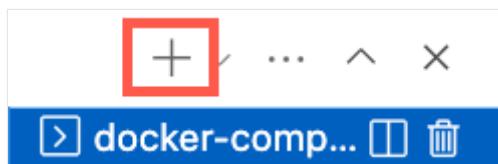
- Mac, Linux, or Windows Subsystem for Linux (WSL):

```
[docker | podman | nerdctl] run --name postgresDb -e POSTGRES_USER=web  
-e POSTGRES_PASSWORD=web-password -e POSTGRES_DB=CustomersDB -v  
$(pwd)/data:/var/lib/postgresql/data -p 5432:5432 postgres
```

- Windows with PowerShell:

```
[docker | podman] run --name postgresDb -e POSTGRES_USER=web -e  
POSTGRES_PASSWORD=web-password -e POSTGRES_DB=CustomersDB -v  
${PWD}/data:/var/lib/postgresql/data -p 5432:5432 postgres
```

4. Once the database container starts, press the + icon in the Visual Studio Code **Terminal toolbar** to create a second terminal window.



5. `cd` into the *server/typescript* folder and run the following commands to install the dependencies and start the API server.

- `npm install`
- `npm start`

6. Press the + icon again in the Visual Studio Code **Terminal toolbar** to create a third terminal window.

7. `cd` into the *client* folder and run the following commands to install the dependencies and start the web server.

- `npm install`
- `npm start`

8. A browser will launch and you'll be taken to <http://localhost:4200>.

Customers

Filter Data:

Company	First Name	Last Name	City	Email	Phone	
Adatum Corporation	Jane	Doe	New York	jane.doe@example.com	+1 (555) 123-4567	Contact Customer
Adventure Works Cycles	John	Smith	London	john.smith@example.com	+1 (555) 123-7654	Contact Customer
Contoso Pharmaceuticals	Peter	Gibbons	Austin	peter.gibbons@example.com	+1 (555) 321-1234	Contact Customer
Tailwind Traders	Lisa	Taylor	Sydney	lisa.taylor@example.com	+1 (555) 123-1234	Contact Customer

Custom Query:

Get the total revenue for all orders. Group by company and include the city.

Run Query

Reset Data

Next Step

AI: Natural Language to SQL

AI: Natural Language to SQL

06/13/2025

The quote "Just because you can doesn't mean you should" is a useful guide when thinking about AI capabilities. For example, Azure OpenAI's natural language to SQL feature allows users to make database queries in plain English, which can be a powerful tool to enhance their productivity. However, *powerful* doesn't always mean *appropriate* or *safe*. This exercise will demonstrate how to use this AI feature while also discussing important considerations to keep in mind before deciding to implement it.

Here's an example of a natural language query that can be used to retrieve data from a database:

```
Get the total revenue for all companies in London.
```

With the proper prompts, Azure OpenAI will convert this query to SQL that can be used to return results from the database. As a result, non-technical users including business analysts, marketers, and executives can more easily retrieve valuable information from databases without grappling with intricate SQL syntax or relying on constrained datagrids and filters. This streamlined approach can boost productivity by eliminating the need for users to seek assistance from technical experts.

This exercise provides a starting point that will help you understand how natural language to SQL works, introduce you to some important considerations, get you thinking about pros and cons, and show you the code to get started.

In this exercise, you will:

- Use GPT prompts to convert natural language to SQL.
- Experiment with different GPT prompts.
- Use the generated SQL to query the PostgreSQL database started earlier.
- Return query results from PostgreSQL and display them in the browser.

Let's start by experimenting with different GPT prompts that can be used to convert natural language to SQL.

Using the Natural Language to SQL Feature

1. In the [previous exercise](#) you started the database, APIs, and application. You also updated the `.env` file. If you didn't complete those steps, follow the instructions at the end of the

exercise before continuing.

2. Go back to the browser (<http://localhost:4200>) and locate the **Custom Query** section of the page below the datagrid. Notice that a sample query value is already included: *Get the total revenue for all orders. Group by company and include the city.*

Custom Query:
Get the total revenue for all orders. Group by company and include the city.

Run Query Reset Data

3. Select the **Run Query** button. This will pass the user's natural language query to Azure OpenAI which will convert it to SQL. The SQL query will then be used to query the database and return any potential results.
4. Run the following **Custom Query**:

Get the total revenue for Adventure Works Cycles. Include the contact information as well.

5. View the terminal window running the API server in Visual Studio Code and notice it displays the SQL query returned from Azure OpenAI. The JSON data is used by the server-side APIs to query the PostgreSQL database. Any string values included in the query are added as parameter values to prevent SQL injection attacks:

JSON

```
{  
  "sql": "SELECT c.company, c.city, c.email, SUM(o.total) AS revenue FROM customers c INNER JOIN orders o ON c.id = o.customer_id WHERE c.company = $1 GROUP BY c.company, c.city, c.email",  
  "paramValues": ["Adventure Works Cycles"]  
}
```

6. Go back to the browser and select **Reset Data** to view all of the customers again in the datagrid.

Exploring the Natural Language to SQL Code

💡 Tip

If you're using Visual Studio Code, you can open files directly by selecting:

- Windows/Linux: `Ctrl + P`
- Mac: `Cmd + P`

Then type the name of the file you want to open.

ⓘ Note

The goal of this exercise is to show what's possible with natural language to SQL functionality and demonstrate how to get started using it. As mentioned earlier, it's important to discuss if this type of AI is appropriate for your organization before proceeding with any implementation. It's also **imperative to plan for proper prompt rules and database security measures** to prevent unauthorized access and protect sensitive data.

1. Now that you've seen the natural language to SQL feature in action, let's examine how it is implemented.
2. Open the `server/apiRoutes.ts` file and locate the `generateSql` route. This API route is called by the client-side application running in the browser and used to generate SQL from a natural language query. Once the SQL query is retrieved, it's used to query the database and return results.

TypeScript

```
router.post('/generateSql', async (req, res) => {
    const userPrompt = req.body.prompt;

    if (!userPrompt) {
        return res.status(400).json({ error: 'Missing parameter "prompt"' });
    }

    try {
        // Call Azure OpenAI to convert the user prompt into a SQL query
        const sqlCommandObject = await getSQLFromNLP(userPrompt);

        let result: any[] = [];
        // Execute the SQL query
        if (sqlCommandObject && !sqlCommandObject.error) {
            result = await queryDb(sqlCommandObject) as any[];
        }
        else {
            result = [ { query_error : sqlCommandObject.error } ];
        }
        res.json(result);
    } catch (e) {
```

```

        console.error(e);
        res.status(500).json({ error: 'Error generating or running SQL
query.' });
    }
});

```

Notice the following functionality in the `generateSql` route:

- It retrieves the user query value from `req.body.prompt` and assigns it to a variable named `userPrompt`. This value will be used in the GPT prompt.
- It calls a `getSQLFromNLP()` function to convert natural language to SQL.
- It passes the generated SQL to a function named `queryDb` that executes the SQL query and returns results from the database.

3. Open the `server/openAI.ts` file in your editor and locate the `getSQLFromNLP()` function. This function is called by the `generatesql` route and is used to convert natural language to SQL.

TypeScript

```

async function getSQLFromNLP(userPrompt: string): Promise<QueryData> {
    // Get the high-level database schema summary to be used in the prompt.
    // The db.schema file could be generated by a background process or the
    // schema could be dynamically retrieved.
    const dbSchema = await fs.promises.readFile('db.schema', 'utf8');

    const systemPrompt = `
        Assistant is a natural language to SQL bot that returns a JSON object
        with the SQL query and
        the parameter values in it. The SQL will query a PostgreSQL database.

        PostgreSQL tables with their columns:
        ${dbSchema}

    Rules:
    - Convert any strings to a PostgreSQL parameterized query value to avoid
    SQL injection attacks.
    - Return a JSON object with the following structure: { "sql": "", "paramValues": [] }

    Examples:

    User: "Display all company reviews. Group by company."
    Assistant: { "sql": "SELECT * FROM reviews", "paramValues": [] }

    User: "Display all reviews for companies located in cities that start
    with 'L'."
    Assistant: { "sql": "SELECT r.* FROM reviews r INNER JOIN customers c ON
    r.customer_id = c.id WHERE c.city LIKE 'L%'", "paramValues": [] }
}

```

```

    User: "Display revenue for companies located in London. Include the
company name and city."
    Assistant: {
        "sql": "SELECT c.company, c.city, SUM(o.total) AS revenue FROM
customers c INNER JOIN orders o ON c.id = o.customer_id WHERE c.city = $1
GROUP BY c.company, c.city",
        "paramValues": ["London"]
    }

    User: "Get the total revenue for Adventure Works Cycles. Include the
contact information as well."
    Assistant: {
        "sql": "SELECT c.company, c.city, c.email, SUM(o.total) AS revenue
FROM customers c INNER JOIN orders o ON c.id = o.customer_id WHERE c.company
= $1 GROUP BY c.company, c.city, c.email",
        "paramValues": ["Adventure Works Cycles"]
    }
};

let queryData: QueryData = { sql: '', paramValues: [], error: '' };
let results = '';

try {
    results = await callOpenAI(systemPrompt, userPrompt);
    if (results) {
        console.log('results', results);
        const parsedResults = JSON.parse(results);
        queryData = { ...queryData, ...parsedResults };
        if (isProhibitedQuery(queryData.sql)) {
            queryData.sql = '';
            queryData.error = 'Prohibited query.';
        }
    }
} catch (error) {
    console.log(error);
    if (isProhibitedQuery(results)) {
        queryData.sql = '';
        queryData.error = 'Prohibited query.';
    } else {
        queryData.error = results;
    }
}

return queryData;
}

```

- A `userPrompt` parameter is passed into the function. The `userPrompt` value is the natural language query entered by the user in the browser.
- A `systemPrompt` defines the type of AI assistant to be used and rules that should be followed. This helps Azure OpenAI understand the database structure, what rules to apply, and how to return the generated SQL query and parameters.

- A function named `callOpenAI()` is called and the `systemPrompt` and `userPrompt` values are passed to it.
- The results are checked to ensure no prohibited values are included in the generated SQL query. If prohibited values are found, the SQL query is set to an empty string.

4. Let's walk through the system prompt in more detail:

TypeScript

```
const systemPrompt = `

Assistant is a natural language to SQL bot that returns a JSON object with
the SQL query and
the parameter values in it. The SQL will query a PostgreSQL database.

PostgreSQL tables with their columns:

${dbSchema}

Rules:
- Convert any strings to a PostgreSQL parameterized query value to avoid
SQL injection attacks.
- Return a JSON object with the following structure: { "sql": "", "paramValues": [] }

Examples:

User: "Display all company reviews. Group by company."
Assistant: { "sql": "SELECT * FROM reviews", "paramValues": [] }

User: "Display all reviews for companies located in cities that start with
'L'." 
Assistant: { "sql": "SELECT r.* FROM reviews r INNER JOIN customers c ON
r.customer_id = c.id WHERE c.city LIKE 'L%'", "paramValues": [] }

User: "Display revenue for companies located in London. Include the company
name and city."
Assistant: {
    "sql": "SELECT c.company, c.city, SUM(o.total) AS revenue FROM customers
c INNER JOIN orders o ON c.id = o.customer_id WHERE c.city = $1 GROUP BY
c.company, c.city",
    "paramValues": ["London"]
}

User: "Get the total revenue for Adventure Works Cycles. Include the
contact information as well."
Assistant: {
    "sql": "SELECT c.company, c.city, c.email, SUM(o.total) AS revenue FROM
customers c INNER JOIN orders o ON c.id = o.customer_id WHERE c.company = $1
GROUP BY c.company, c.city, c.email",
    "paramValues": ["Adventure Works Cycles"]
```

```
}
```

- The type of AI assistant to be used is defined. In this case a "natural language to SQL bot".
- Table names and columns in the database are defined. The high-level schema included in the prompt can be found in the *server/db.schema* file and looks like the following.

```
- customers (id, company, city, email)
- orders (id, customer_id, date, total)
- order_items (id, order_id, product_id, quantity, price)
- reviews (id, customer_id, review, date, comment)
```

💡 Tip

You may consider creating read-only views that only contain the data users are allowed to query using natural language to SQL.

- A rule is defined to convert any string values to a parameterized query value to avoid SQL injection attacks.
- A rule is defined to always return a JSON object with the SQL query and the parameter values in it.
- Example user prompts and the expected SQL query and parameter values are provided. This is referred to as "[few-shot learning](#)". Although LLMs are trained on large amounts of data, they can be adapted to new tasks with only a few examples. An alternative approach is "zero-shot" learning where no example is provided and the model is expected to generate the correct SQL query and parameter values.

5. The `getSQLFromNLP()` function sends the system and user prompts to a function named `callOpenAI()` which is also located in the *server/openAI.ts* file. The `callOpenAI()` function determines if the Azure OpenAI service or OpenAI service should be called by checking environment variables. If a key, endpoint, and model are available in the environment variables then Azure OpenAI is called, otherwise OpenAI is called.

TypeScript

```
function callOpenAI(systemPrompt: string, userPrompt: string, temperature = 0, useBYOD = false) {
```

```
const isAzureOpenAI = OPENAI_API_KEY && OPENAI_ENDPOINT && OPENAI_MODEL;

if (isAzureOpenAI) {
    if (useBYOD) {
        return getAzureOpenAIBYODCompletion(systemPrompt, userPrompt,
temperature);
    }
    return getAzureOpenAICompletion(systemPrompt, userPrompt,
temperature);
}

return getOpenAICompletion(systemPrompt, userPrompt, temperature);
}
```

ⓘ Note

Although we'll focus on Azure OpenAI throughout this tutorial, if you only supply an `OPENAI_API_KEY` value in the `.env` file, the application will use OpenAI instead. If you choose to use OpenAI instead of Azure OpenAI you may see different results in some cases.

6. Locate the `getAzureOpenAICompletion()` function.

TypeScript

```
async function getAzureOpenAICompletion(systemPrompt: string, userPrompt:
string, temperature: number): Promise<string> {
    const completion = await createAzureOpenAICompletion(systemPrompt,
userPrompt, temperature);
    let content = completion.choices[0]?.message?.content?.trim() ?? '';
    console.log('Azure OpenAI Output: \n', content);
    if (content && content.includes('{') && content.includes('}')) {
        content = extractJson(content);
    }
    return content;
}
```

This function does the following:

- **Parameters:**
 - `systemPrompt`, `userPrompt`, and `temperature` are the main parameters.
 - `systemPrompt`: Informs the Azure OpenAI model of its role and the rules to follow.
 - `userPrompt`: Contains the user-provided information such as natural language input or rules for generating the output.

- `temperature`: Dictates the creativity level of the model's response. A higher value results in more creative outputs, while lower values (e.g., 0) produce more deterministic answers.
- **Completion Generation:**
 - The function calls `createAzureOpenAICompletion()` with `systemPrompt`, `userPrompt`, and `temperature` to generate a completion.
 - It extracts the content from the first choice in the completion, trimming any extra whitespace.
 - If the content contains JSON-like structures (indicated by the presence of `{` and `}`), it extracts the JSON content.
- **Logging and Return Value:**
 - The function logs the Azure OpenAI output to the console.
 - It returns the processed content as a string.

7. Locate the `createAzureOpenAICompletion()` function.

TypeScript

```
async function createAzureOpenAICompletion(systemPrompt: string, userPrompt: string, temperature: number, dataSources?: any[]): Promise<any> {
    const baseEnvVars = ['OPENAI_API_KEY', 'OPENAI_ENDPOINT', 'OPENAI_MODEL'];
    const byodEnvVars = ['AZURE_AI_SEARCH_ENDPOINT', 'AZURE_AI_SEARCH_KEY', 'AZURE_AI_SEARCH_INDEX'];
    const requiredEnvVars = dataSources ? [...baseEnvVars, ...byodEnvVars] : baseEnvVars;
    checkRequiredEnvVars(requiredEnvVars);

    const config = {
        apiKey: OPENAI_API_KEY,
        endpoint: OPENAI_ENDPOINT,
        apiVersion: OPENAI_API_VERSION,
        deployment: OPENAI_MODEL
    };
    const aoai = new AzureOpenAI(config);
    const completion = await aoai.chat.completions.create({
        model: OPENAI_MODEL, // gpt-4o, gpt-3.5-turbo, etc. Pulled from .env
        file
        max_tokens: 1024,
        temperature,
        response_format: {
            type: "json_object",
        },
        messages: [
            { role: 'system', content: systemPrompt },
            { role: 'user', content: userPrompt }
        ],
    });
}
```

```

        // @ts-expect-error data_sources is a custom property used with the
        "Azure Add Your Data" feature
        data_sources: dataSources
    });
    return completion;
}

function checkRequiredEnvVars(requiredEnvVars: string[]) {
    for (const envVar of requiredEnvVars) {
        if (!process.env[envVar]) {
            throw new Error(`Missing ${envVar} in environment variables.`);
        }
    }
}

```

This function does the following:

- **Parameters:**
 - `systemPrompt`, `userPrompt`, and `temperature` are the main parameters discussed earlier.
 - An optional `dataSources` parameter supports the "Azure Bring Your Own Data" feature, which will be covered later in this tutorial.
- **Environment Variables Check:**
 - The function verifies the presence of essential environment variables, throwing an error if any are missing.
- **Configuration Object:**
 - A `config` object is created using values from the `.env` file (`OPENAI_API_KEY`, `OPENAI_ENDPOINT`, `OPENAI_API_VERSION`, `OPENAI_MODEL`). These values are used to construct the URL for calling Azure OpenAI.
- **AzureOpenAI Instance:**
 - An instance of `AzureOpenAI` is created using the `config` object. The `AzureOpenAI` symbol is part of the `openai` package, which should be imported at the top of your file.
- **Generating a Completion:**
 - The `chat.completions.create()` function is called with the following properties:
 - `model`: Specifies the GPT model (e.g., `gpt-4o`, `gpt-3.5-turbo`) as defined in your `.env` file.
 - `max_tokens`: Defines the maximum number of tokens for the completion.
 - `temperature`: Sets the sampling temperature. Higher values (e.g., `0.9`) yield more creative responses, while lower values (e.g., `0`) produce more deterministic answers.

- `response_format`: Defines the response format. Here, it's set to return a JSON object. More details on JSON mode can be found in the [Azure OpenAI reference documentation](#).
 - `messages`: Contains the messages for generating chat completions. This example includes two messages: one from the system (defining behavior and rules) and one from the user (containing the prompt text).
- **Return Value:**
 - The function returns the completion object generated by Azure OpenAI.

8. Comment out the following lines in the `getSQLFromNLP()` function:

TypeScript

```
// if (isProhibitedQuery(queryData.sql)) {  
//     queryData.sql = '';  
// }
```

9. Save `openAI.ts`. The API server will automatically rebuild the TypeScript code and restart the server.

10. Go back to the browser and enter *Select all table names from the database* into the **Custom Query** input. Select **Run Query**. Are table names displayed?

11. Go back to the `getSQLFromNLP()` function in `server/openAI.ts` and add the following rule into the `Rules:` section of the system prompt and then save the file.

- Do not allow the SELECT query to return table names, function names, or procedure names.

12. Go back to the browser and perform the following tasks:

- Enter *Select all table names from the database* into the **Custom Query** input. Select **Run Query**. Are table names displayed?
- Enter *Select all function names from the database*. into the **Custom Query** input and select **Run Query** again. Are function names displayed?

13. QUESTION: Will a model always follow the rules you define in the prompt?

ANSWER: No! It's important to note that OpenAI models can return unexpected results on occasion that may not match the rules you've defined. It's important to plan for that in your code.

14. Go back to `server/openAI.ts` and locate the `isProhibitedQuery()` function. This is an example of post-processing code that can be run after Azure OpenAI returns results. Notice that it sets the `sql` property to an empty string if prohibited keywords are returned in the generated SQL query. This ensures that if unexpected results are returned from Azure OpenAI, the SQL query will not be run against the database.

TypeScript

```
function isProhibitedQuery(query: string): boolean {
    if (!query) return false;

    const prohibitedKeywords = [
        'insert', 'update', 'delete', 'drop', 'truncate', 'alter', 'create',
        'replace',
        'information_schema', 'pg_catalog', 'pg_tables', 'pg_proc',
        'pg_namespace', 'pg_class',
        'table_schema', 'table_name', 'column_name', 'column_default',
        'is_nullable',
        'data_type', 'udt_name', 'character_maximum_length',
        'numeric_precision',
        'numeric_scale', 'datetime_precision', 'interval_type',
        'collation_name',
        'grant', 'revoke', 'rollback', 'commit', 'savepoint', 'vacuum',
        'analyze'
    ];
    const queryLower = query.toLowerCase();
    return prohibitedKeywords.some(keyword => queryLower.includes(keyword));
}
```

(!) Note

It's important to note that this is only demo code. There may be other prohibited keywords required to cover your specific use cases if you choose to convert natural language to SQL. This is a feature that you must plan for and use with care to ensure that only valid SQL queries are returned and run against the database. In addition to prohibited keywords, you will also need to factor in security as well.

15. Go back to `server/openAI.ts` and uncomment the following code in the `getSQLFromNLP()` function. Save the file.

TypeScript

```
if (isProhibitedQuery(queryData.sql)) {
    queryData.sql = '';
}
```

16. Remove the following rule from `systemPrompt` and save the file.

- Do not allow the SELECT query to return table names, function names, or procedure names.

17. Go back to the browser, enter *Select all table names from the database* into the **Custom Query** input again and select the **Run Query** button.

18. Do any table results display? Even without the rule in place, the `isProhibitedQuery` post-processing code prohibits that type of query from being run against the database.

19. As discussed earlier, integrating natural language to SQL in line of business applications can be quite beneficial to users, but it does come with its own set of considerations.

Advantages:

- User-friendliness: This feature can make database interaction more accessible to users without technical expertise, reducing the need for SQL knowledge and potentially speeding up operations.
- Increased productivity: Business analysts, marketers, executives, and other non-technical users can retrieve valuable information from databases without having to rely on technical experts, thereby increasing efficiency.
- Broad application: By using advanced language models, applications can be designed to cater to a wide range of users and use-cases.

Considerations:

- Security: One of the biggest concerns is security. If users can interact with databases using natural language, there needs to be robust security measures in place to prevent unauthorized access or malicious queries. You may consider implementing a read-only mode to prevent users from modifying data.
- Data Privacy: Certain data might be sensitive and should not be easily accessible, so you'll need to ensure proper safeguards and user permissions are in place.
- Accuracy: While natural language processing has improved significantly, it's not perfect. Misinterpretation of user queries could lead to inaccurate results or unexpected behavior. You'll need to plan for how unexpected results will be handled.

- Efficiency: There are no guarantees that the SQL returned from a natural language query will be efficient. In some cases, additional calls to Azure OpenAI may be required if post-processing rules detect issues with SQL queries.
- Training and User Adaptation: Users need to be trained to formulate their queries correctly. While it's easier than learning SQL, there can still be a learning curve involved.

20. A few final points to consider before moving on to the next exercise:

- Remember that, "Just because you can doesn't mean you should" applies here. Use extreme caution and careful planning before integrating natural language to SQL into an application. It's important to understand the potential risks and to plan for them.
- Before using this type of technology, discuss potential scenarios with your team, database administrators, security team, stakeholders, and any other relevant parties to ensure that it's appropriate for your organization. It's important to discuss if natural language to SQL meets security, privacy, and any other requirements your organization may have in place.
- Security should be a primary concern and built into the planning, development, and deployment process.
- While natural language to SQL can be very powerful, careful planning must go into it to ensure prompts have required rules and that post-processing functionality is included. Plan for additional time to implement and test this type of functionality and to account for scenarios where unexpected results are returned.
- With Azure OpenAI, customers get the security capabilities of Microsoft Azure while running the same models as OpenAI. Azure OpenAI offers private networking, regional availability, and responsible AI content filtering. Learn more about [Data, privacy, and security for Azure OpenAI Service](#).

21. You've now seen how to use Azure OpenAI to convert natural language to SQL and learned about the pros and cons of implementing this type of functionality. In the next exercise, you'll learn how email and SMS messages can be generated using Azure OpenAI.

Next Step

[AI: Generating Completions](#)

AI: Generating Completions

06/13/2025

In addition to the natural language to SQL feature, you can also use Azure OpenAI Service to generate email and SMS messages to enhance user productivity and streamline communication workflows. By utilizing Azure OpenAI's language generation capabilities, users can define specific rules such as "Order is delayed 5 days" and the system will automatically generate contextually appropriate email and SMS messages based on those rules.

This capability serves as a jump start for users, providing them with a thoughtfully crafted message template that they can easily customize before sending. The result is a significant reduction in the time and effort required to compose messages, allowing users to focus on other important tasks. Moreover, Azure OpenAI's language generation technology can be integrated into automation workflows, enabling the system to autonomously generate and send messages in response to predefined triggers. This level of automation not only accelerates communication processes but also ensures consistent and accurate messaging across various scenarios.

In this exercise, you will:

- Experiment with different prompts.
- Use prompts to generate completions for email and SMS messages.
- Explore code that enables AI completions.
- Learn about the importance of prompt engineering and including rules in your prompts.

Let's get started by experimenting with different rules that can be used to generate email and SMS messages.

Using the AI Completions Feature

1. In a [previous exercise](#) you started the database, APIs, and application. You also updated the `.env` file. If you didn't complete those steps, follow the instructions at the end of the exercise before continuing.
2. Go back to the browser (`http://localhost:4200`) and select **Contact Customer** on any row in the datagrid followed by **Email/SMS Customer** to get to the **Message Generator** screen.
3. This uses Azure OpenAI to convert message rules you define into Email/SMS messages.
Perform the following tasks:
 - Enter a rule such as *Order is delayed 5 days* into the input and select the **Generate Email/SMS Messages** button.

Contact Adatum Corporation

Message Generator

Email/SMS

Message Rules:

Order is delayed 5 days

Generate Email/SMS Messages

- You will see a subject and body generated for the email and a short message generated for the SMS.

ⓘ Note

Because Azure Communication Services isn't enabled yet, you won't be able to send the email or SMS messages.

4. Close the email/SMS dialog window in the browser. Now that you've seen this feature in action, let's examine how it's implemented.

Exploring the AI Completions Code

💡 Tip

If you're using Visual Studio Code, you can open files directly by selecting:

- Windows/Linux: `Ctrl + P`
- Mac: `Cmd + P`

Then type the name of the file you want to open.

1. Open the `server/apiRoutes.ts` file and locate the `completeEmailSmsMessages` route. This API is called by front-end portion of the app when the **Generate Email/SMS Messages** button is selected. It retrieves the user prompt, company, and contact name values from the body and passes them to the `completeEmailSMSMessages()` function in the `server/openAI.ts` file. The results are then returned to the client.

TypeScript

```
router.post('/completeEmailSmsMessages', async (req, res) => {
  const { prompt, company, contactName } = req.body;

  if (!prompt || !company || !contactName) {
    return res.status(400).json({
      status: false,
      error: 'The prompt, company, and contactName parameters must be provided.'
    });
  }

  let result;
  try {
    // Call OpenAI to get the email and SMS message completions
    result = await completeEmailSMSMessages(prompt, company, contactName);
  }
  catch (e: unknown) {
    console.error('Error parsing JSON:', e);
  }

  res.json(result);
});
```

2. Open the `server/openAI.ts` file and locate the `completeEmailSMSMessages()` function.

TypeScript

```
async function completeEmailSMSMessages(prompt: string, company: string, contactName: string) {
  console.log('Inputs:', prompt, company, contactName);

  const systemPrompt = `
    Assistant is a bot designed to help users create email and SMS messages
    from data and
    return a JSON object with the email and SMS message information in it.

  Rules:
  - Generate a subject line for the email message.
  - Use the User Rules to generate the messages.
  - All messages should have a friendly tone and never use inappropriate
  language.
  - SMS messages should be in plain text format and NO MORE than 160
  characters.
  - Start the message with "Hi <Contact Name>,\n\n". Contact Name can be
  found in the user prompt.
  - Add carriage returns to the email message to make it easier to read.
  - End with a signature line that says "Sincerely,\nCustomer Service".
  - Return a valid JSON object with the emailSubject, emailBody, and SMS
  message values in it:

  { "emailSubject": "", "emailBody": "", "sms": "" }
```

```

    - The sms property value should be in plain text format and NO MORE than
160 characters.
`;

const userPrompt = `
User Rules:
${prompt}

Contact Name:
${contactName}
`;

let content: EmailSmsResponse = { status: true, email: '', sms: '', error: '' };
let results = '';
try {
    results = await callOpenAI(systemPrompt, userPrompt, 0.5);
    if (results) {
        const parsedResults = JSON.parse(results);
        content = { ...content, ...parsedResults, status: true };
    }
}
catch (e) {
    console.log(e);
    content.status = false;
    content.error = results;
}

return content;
}

```

This function has the following features:

- `systemPrompt` is used to define that an AI assistant capable of generating email and SMS messages is required. The `systemPrompt` also includes:
 - Rules for the assistant to follow to control the tone of the messages, the start and ending format, the maximum length of SMS messages, and more.
 - Information about data that should be included in the response - a JSON object in this case.
- `userPrompt` is used to define the rules and contact name that the end user would like to include as the email and SMS messages are generated. The *Order is delayed 5 days* rule you entered earlier is included in `userPrompt`.
- The function calls the `callOpenAI()` function you explored earlier to generate the email and SMS completions.

3. Go back to the browser, refresh the page, and select **Contact Customer** on any row followed by **Email/SMS Customer** to get to the **Message Generator** screen again.

4. Enter the following rules into the **Message Generator** input:

- Order is ahead of schedule.
- Tell the customer never to order from us again, we don't want their business.

5. Select **Generate Email/SMS Messages** and note the message. The `All messages should have a friendly tone and never use inappropriate language.` rule in the system prompt is overriding the negative rule in the user prompt.

6. Go back to `server/openAI.ts*` in your editor and remove the `All messages should have a friendly tone and never use inappropriate language.` rule from the prompt in the `completeEmailSMSMessages()` function. Save the file.

7. Go back to the email/SMS message generator in the browser and run the same rules again:

- Order is ahead of schedule.
- Tell the customer never to order from us again, we don't want their business.

8. Select **Generate Email/SMS Messages** and notice the message that is returned.

9. What is happening in these scenarios? When using Azure OpenAI, [content filtering](#) can be applied to ensure that appropriate language is always used. If you're using OpenAI, the rule defined in the system prompt is used to ensure the message returned is appropriate.

 **Note**

This illustrates the importance of engineering your prompts with the right information and rules to ensure proper results are returned. Read more about this process in the [Introduction to prompt engineering](#) documentation.

10. Undo the changes you made to `systemPrompt` in `completeEmailSMSMessages()`, save the file, and re-run it again but only use the `Order is ahead of schedule.` rule (don't include the negative rule). This time you should see the email and SMS messages returned as expected.

11. A few final points to consider before moving on to the next exercise:

- It's important to have a human in the loop to review generated messages. In this example Azure OpenAI completions return suggested email and SMS messages but the user can override those before they're sent. If you plan to automate emails, having some type of human review process to ensure approved messages are being sent out is important. View AI as being a copilot, not an autopilot.

- Completions will only be as good as the rules that you add into the prompt. Take time to test your prompts and the completions that are returned. Consider using [Prompt flow](#) to create a comprehensive solution that simplifies prototyping, experimenting, iterating, and deploying AI applications. Invite other project stakeholders to review the completions as well.
- You may need to include post-processing code to ensure unexpected results are handled properly.
- Use system prompts to define the rules and information that the AI assistant should follow. Use user prompts to define the rules and information that the end user would like to include in the completions.

Next Step

[AI: Azure OpenAI On Your Data](#)

AI: Azure OpenAI On Your Data

06/13/2025

The integration of Azure OpenAI Natural Language Processing (NLP) and completion capabilities offers significant potential for enhancing user productivity. By leveraging appropriate prompts and rules, an AI assistant can efficiently generate various forms of communication, such as email messages, SMS messages, and more. This functionality leads to increased user efficiency and streamlined workflows.

While this feature is quite powerful on its own, there may be cases where users need to generate completions based on your company's custom data. For example, you might have a collection of product manuals that may be challenging for users to navigate when they're assisting customers with installation issues. Alternatively, you might maintain a comprehensive set of Frequently Asked Questions (FAQs) related to healthcare benefits that can prove challenging for users to read through and get the answers they need. In these cases and many others, Azure OpenAI Service enables you to leverage your own data to generate completions, ensuring a more tailored and contextually accurate response to user questions.

Here's a quick overview of how the "bring your own data" feature works from the [Azure OpenAI documentation](#).

ⓘ Note

One of the key features of Azure OpenAI on your data is its ability to retrieve and utilize data in a way that enhances the model's output. Azure OpenAI on your data, together with Azure AI Search, determines what data to retrieve from the designated data source based on the user input and provided conversation history. This data is then augmented and resubmitted as a prompt to the OpenAI model, with retrieved information being appended to the original prompt. Although retrieved data is being appended to the prompt, the resulting input is still processed by the model like any other prompt. Once the data has been retrieved and the prompt has been submitted to the model, the model uses this information to provide a completion.

In this exercise, you will:

- Create a custom data source using the Azure AI Foundry portal.
- Deploy an embedding model using the Azure AI Foundry portal.
- Upload custom documents.
- Start a chat session in the Chat playground to experiment with generating completions based upon your own data.

- Explore code that uses Azure AI Search and Azure OpenAI to generate completions based upon your own data.

Let's get started by deploying an embedding model and adding a custom data source in Azure AI Foundry.

Adding a Custom Data Source to Azure AI Foundry

1. Navigate to [Azure OpenAI Studio](#) and sign in with credentials that have access to your Azure OpenAI resource.
2. Select **Deployments** from the navigation menu.
3. Select **Select Deploy model --> Deploy base model** in the toolbar.
4. Select the **text-embedding-ada-002** model from the list of models and select **Confirm**.
5. Select the following options:
 - Deployment name: **text-embedding-ada-002**
 - Model version: **Default**
 - Deployment type: **Standard**
 - Set the **Tokens per Minute Rate Limit (thousands)** value to **120K**
 - Content Filter: **DefaultV2**
 - Enable dynamic quote: **Enabled**
6. Select the **Deploy** button.
7. After the model is created, select **Home** from the navigation menu to go to the welcome screen.
8. Locate the **Bring your own data** tile on the welcome screen and select **Try it now**.

Welcome to Azure OpenAI service

Explore the generative AI models and craft unique prompts for your use cases.

Get started

Assistants playground

Speed up development of GPT-powered AI Assistants with prebuilt conversation state management and customization tools.

[Try it now](#)

Chat playground

Design a customized AI assistant using ChatGPT. Experiment with GPT-3.5-Turbo and GPT-4 models.

[Try it now](#)

Bring your own data

Ground your own data on advanced AI models to create conversational copilots that aid user comprehension, task completion, and decision-making.

[Try it now](#)

Completions playground

Experiment with completions models for use cases such as summarization, content generation, and classification.

[Try it now](#)

Images playground

Generate unique images by writing descriptions in natural language.

[Try it now](#)

9. Select **Add your data** followed by **Add a data source**.

10. In the **Select data source** dropdown, select **Upload files**.

11. Under the **Select Azure Blob storage resource** dropdown, select **Create a new Azure Blob storage resource**.

12. Select your Azure subscription in the **Subscription** dropdown.

13. Under the **Select Azure Blob storage resource** dropdown, select **Create a new Azure Blob storage resource**.

14. This will take you to the Azure portal where you can perform the following tasks:

- Enter a unique name for the storage account such as **byodstorage[Your Last Name]**.
- Select a region that's close to your location.
- Select **Review** followed by **Create**.

15. Once the blob storage resource is created, go back to the Azure AI Foundry dialog and select your newly created blob storage resource from the **Select Azure Blob storage resource** dropdown. If you don't see it listed, select the refresh icon next to the dropdown.

16. Cross-origin resource sharing (CORS) needs to be turned on in order for your storage account to be accessed. Select **Turn on CORS** in the Azure AI Foundry dialog.

ⓘ Azure OpenAI needs your permission to access this resource

In order for Azure OpenAI to access the storage account in your Azure subscription, your permission is needed for security reasons. This is known as [Cross-origin resource sharing \(CORS\)](#). If you do not want to turn on CORS, you'll need to choose a different storage account or select a different data source to proceed.

[Turn on CORS](#)

17. Under the **Select Azure AI Search resource** dropdown, select **Create a new Azure AI Search resource**.

18. This will take you back to the Azure portal where you can perform the following tasks:

- Enter a unique name for the AI Search resource such as **byodsearch-[Your Last Name]**.
- Select a region that's close to your location.
- In the **Pricing tier** section, select **Change Pricing Tier** and select **Basic** followed by **Select**. The free tier isn't supported, so you'll clean up the AI Search resource at the end of this tutorial.
- Select **Review** followed by **Create**.

19. Once the AI Search resource is created, go to the resource **Overview** page and copy the **Url** value to a local file.

The screenshot shows the Azure AI Search resource Overview page. At the top, there are navigation icons: Add index, Import data, Search explorer, Refresh, Delete, Move, and a dropdown. Below this is a section titled 'Essentials' with the following details:

Resource group (move)	: Sandbox	Url	:
Location (move)	: West US 2	Pricing tier	: Basic
Subscription (move)	: Visual Studio Enterprise Subscription	Replicas	: 1 (No SLA)
Subscription ID	:	Partitions	: 1
Status	: Running	Search units	: 1
Tags (edit)	: ProjectType : aoai-your-data-service		

20. Select **Settings --> Keys** in the navigation menu.

21. On the **API Access control** page, select **Both** to enable the service to be accessed by using Managed Identity or by using a key. Select **Yes** when prompted.

ⓘ Note

Although we'll use an API key in this exercise since adding role assignments can take up to 10 minutes, with a little additional effort you can enable a system assigned managed identity to access the service more securely.

22. Select **Keys** in the left navigation menu and copy the **Primary admin key** value to a local file. You'll need the URL and key values later in the exercise.

23. Select **Settings --> Semantic ranker** in the navigation menu and ensure that **Free** is selected.

 **Note**

To check if semantic ranker is available in a specific region, [Check the Products Available by Region](#) page on the Azure web site to see if your region is listed.

24. Go back to the Azure AI Foundry **Add Data** dialog and select your newly created search resource from the **Select Azure AI Search resource** dropdown. If you don't see it listed, select the refresh icon next to the dropdown.

25. Enter a value of **byod-search-index** for the **Enter the index name** value.

26. Select the **Add vector search to this search resource** checkbox.

27. In the **Select an embedding model** dropdown, select the **text-embedding-ada-002** model you created earlier.

28. In the **Upload files** dialog, select **Browse for a file**.

29. Navigate to the project's *customer documents* folder (located at the root of the project) and select the following files:

- *Clock A102 Installation Instructions.docx*
- *Company FAQs.docx*

 **Note**

This feature currently supports the following file formats for local index creation: .txt, .md, .html, .pdf, .docx, and .pptx.

30. Select **Upload files**. The files will be uploaded into a **fileupload-byod-search-index** container in the blob storage resource you created earlier.

31. Select **Next** to go to the **Data management** dialog.

32. In the **Search type** dropdown, select **Hybrid + semantic**.

 **Note**

This option provides support for keyword and vector search. Once results are returned, a secondary ranking process is applied to the result set using deep learning models which improves the search relevance for the user. To learn more about semantic search, view the [Semantic search in Azure AI Search](#) documentation.

33. Ensure that the **Select a size** value is set to **1024**.
34. Select **Next**.
35. For the **Azure resource authentication type**, select **API key**. Learn more about selecting the right authentication type in the [Azure AI Search authentication documentation](#).
36. Select **Next**.
37. Review the details and select **Save and close**.
38. Now that your custom data has been uploaded, the data will be indexed and available to use in the **Chat playground**. This process may take a few minutes. Once it's completed, continue to the next section.

Using Your Custom Data Source in the Chat Playground

1. Locate the **Chat session** section of the page in Azure OpenAI Studio and enter the following **User query**:

text

What safety rules are required to install a clock?

2. After submitting the user query you should see a result similar to the following displayed:

 Clear chat

 Chat capabilities



Show JSON



What safety rules are required to install a clock?

...  

The safety rules required to install a clock include the following:

1. **Work Environment:** Ensure you work in a well-lit area with a clear workspace to avoid accidents¹.
2. **Handling Components:** Handle small clock components with care to prevent injury or damage¹.
3. **Safety Gear:** Wear appropriate protective gear, such as safety glasses and gloves¹.
4. **Children and Pets:** Keep children and pets away from the assembly area to avoid any potential accidents¹.

> 1 references

Type user query here. (Shift + Enter for new line)

126/128000 tokens to be sent 

3. Expand the **1 references** section in the chat response and notice that the *Clock A102 Installation Instructions.docx* file is listed and that you can select it to view the document.

4. Enter the following **User message**:

text

What should I do to mount the clock on the wall?

5. You should see a result similar to the following displayed:

What should I do to mount the clock on the wall?

... ⌂ ⌂

To mount the Clock A102 on the wall, follow these steps:

1. **Determine the Desired Location:** Choose a spot on the wall where the clock will be visible, and consider the weight-bearing capacity and room decor¹.
2. **Locate Wall Studs or Anchors:** Use a stud finder or electronic detector to locate wall studs or anchors for added stability¹.
3. **Mark the Mounting Holes:** Mark the positions of the mounting holes on the wall, ensuring proper alignment¹.
4. **Create Mounting Holes:** Using a drill and the appropriate drill bits, create holes for the mounting screws or anchors at the marked positions¹.
5. **Attach the Clock to the Wall:** Securely attach the clock to the wall using the provided mounting hardware, ensuring it is level and securely fastened¹.

By following these steps, you will ensure a secure and aesthetically pleasing installation of your clock.

› 1 references

6. Now let's experiment with the Company FAQs document. Enter the following text into the **User query** field:

text

What is the company's policy on vacation time?

7. You should see that no information was found for that request.

8. Enter the following text into the **User query** field:

text

How should I handle refund requests?

9. You should see a result similar to the following displayed:

...

If a company requests a refund or wishes to return a product, follow the established refund policy guidelines. You should:

1. Gather the necessary details, such as the reason for the request and the condition of the product.
2. Communicate with the company to provide a suitable resolution¹.

> 1 references

10. Expand the **1 references** section in the chat response and notice that the *Company FAQs.docx* file is listed and that you can select it to view the document.

11. Select **View code** in the toolbar of the **Chat playground**.

Chat playground

Export ▾

View Code

Deploy ▾

Import ▾

Prompt samples

12. Note that you can switch between different languages, view the endpoint, and access the endpoint's key. Close the **Sample Code** dialog window.

Sample Code

You can use the following code to start integrating your current prompt and settings into your application.

Endpoint ⓘ

<https://danwahlin-ai-test.openai.azure.com/>



API key ⓘ

.....



Azure Search Resource Key ⓘ

.....



You should use environment variables or a secret management tool like Azure Key Vault to prevent accidental exposure of your key in applications.

[Learn more about setting up an environment](#)

javascript ▾

```
1 import { AzureOpenAI } from "openai";
2 import { DefaultAzureCredential, getBearerTokenProvider } from "msal";
3
4 export async function main() {
5   const scope = "https://cognitiveservices.azure.com/.default";
6   const azureADTokenProvider = getBearerTokenProvider(new DefaultAzureCredential());
7   const deployment = "gpt-35-turbo";
8   const apiVersion = "2024-04-01-preview";
9   const client = new AzureOpenAI({ azureADTokenProvider, deployment });
10  const result = await client.chat.completions.create({
11    messages: [
12      { role: "system", content: "You are a helpful assistant. You will answer all questions accurately and provide helpful information." },
13      { role: "user", content: "Can you help me?" },
14    ],
15    model: '',
16  });
17
18  for (const choice of result.choices) {
19    console.log(choice.message);
20  }
21}
22
23 main().catch((err) => {
24   console.error("The sample encountered an error:", err);
25});
```

Close

13. Turn on the **Show raw JSON** toggle above the chat messages. Notice the chat session starts with a message similar to the following:

JSON

```
{  
  "role": "system",  
  "content": "You are an AI assistant that helps people find information."  
}
```

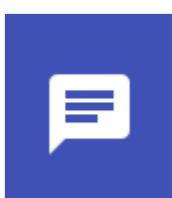
14. Now that you've created a custom data source and experimented with it in the Chat playground, let's see how you can use it in the project's application.

Using the Bring Your Own Data Feature in the Application

1. Go back to the project in VS Code and open the `.env` file. Update the following values with your AI Services endpoint, key, and index name. You copied the endpoint and key to a local file earlier in this exercise.

```
AZURE_AI_SEARCH_ENDPOINT=<AI_SERVICES_ENDPOINT_VALUE>  
AZURE_AI_SEARCH_KEY=<AI_SERVICES_KEY_VALUE>  
AZURE_AI_SEARCH_INDEX=byod-search-index
```

2. In a [previous exercise](#) you started the database, APIs, and application. You also updated the `.env` file. If you didn't complete those steps, follow the instructions at the end of the earlier exercise before continuing.
3. Once the application has loaded in the browser, select the **Chat Help** icon in the upper-right of the application.



4. The following text should appear in the chat dialog:

text

How should I handle a company refund request?

5. Select the **Get Help** button. You should see results returned from the *Company FAQs.docx* document that you uploaded earlier in Azure OpenAI Studio. If you'd like to read through the document, you can find it in the **customer documents** folder at the root of the project.

6. Change the text to the following and select the **Get Help** button:

```
text  
  
What safety rules are required to install a clock?
```

7. You should see results returned from the *Clock A102 Installation Instructions.docx* document that you uploaded earlier in Azure OpenAI Studio. This document is also available in the **customer documents** folder at the root of the project.

Exploring the Code

💡 Tip

If you're using Visual Studio Code, you can open files directly by selecting:

- Windows/Linux: `Ctrl + P`
- Mac: `Cmd + P`

Then type the name of the file you want to open.

1. Go back to the project source code in Visual Studio Code.

2. Open the *server/apiRoutes.ts* file and locate the `completeBYOD` route. This API is called when the **Get Help** button is selected in the Chat Help dialog. It retrieves the user prompt from the request body and passes it to the `completeBYOD()` function in the *server/openAI.ts* file. The results are then returned to the client.

```
TypeScript  
  
router.post('/completeBYOD', async (req, res) => {  
    const { prompt } = req.body;  
  
    if (!prompt) {  
        return res.status(400).json({  
            status: false,  
            error: 'The prompt parameter must be provided.'  
        });  
    }  
}
```

```

let result;
try {
    // Call OpenAI to get custom "bring your own data" completion
    result = await completeBYOD(prompt);
}
catch (e: unknown) {
    console.error('Error parsing JSON:', e);
}

res.json(result);
});

```

3. Open the `server/openAI.ts` file and locate the `completeBYOD()` function.

TypeScript

```

async function completeBYOD(userPrompt: string): Promise<string> {
    const systemPrompt = 'You are an AI assistant that helps people find
information in documents.';
    return await callOpenAI(systemPrompt, userPrompt, 0, true);
}

```

This function has the following features:

- The `userPrompt` parameter contains the information the user typed into the chat help dialog.
- the `systemPrompt` variable defines that an AI assistant designed to help people find information will be used.
- `callOpenAI()` is used to call the Azure OpenAI API and return the results. It passes the `systemPrompt` and `userPrompt` values as well as the following parameters:
 - `temperature` - The amount of creativity to include in the response. The user needs consistent (less creative) answers in this case so the value is set to 0.
 - `useBYOD` - A boolean value that indicates whether or not to use AI Search along with Azure OpenAI. In this case, it's set to `true` so AI Search functionality will be used.

4. The `callOpenAI()` function accepts a `useBYOD` parameter that is used to determine which OpenAI function to call. In this case, it sets `useBYOD` to `true` so the `getAzureOpenAIBYODCompletion()` function will be called.

TypeScript

```

function callOpenAI(systemPrompt: string, userPrompt: string, temperature =
0, useBYOD = false) {
    const isAzureOpenAI = OPENAI_API_KEY && OPENAI_ENDPOINT && OPENAI_MODEL;

```

```

    if (isAzureOpenAI) {
        if (useBYOD) {
            return getAzureOpenAIBYODCompletion(systemPrompt, userPrompt,
temperature);
        }
        return getAzureOpenAICompletion(systemPrompt, userPrompt,
temperature);
    }

    return getOpenAICompletion(systemPrompt, userPrompt, temperature);
}

```

5. Locate the `getAzureOpenAIBYODCompletion()` function in `server/openAI.ts`. It's quite similar to the `getAzureOpenAICompletion()` function you examined earlier, but is shown as a separate function to highlight a few key differences that are unique to the "Azure OpenAI on your data" scenario available in Azure OpenAI.

TypeScript

```

async function getAzureOpenAIBYODCompletion(systemPrompt: string, userPrompt:
string, temperature: number): Promise<string> {
    const dataSources = [
        {
            type: 'azure_search',
            parameters: {
                authentication: {
                    type: 'api_key',
                    key: AZURE_AI_SEARCH_KEY
                },
                endpoint: AZURE_AI_SEARCH_ENDPOINT,
                index_name: AZURE_AI_SEARCH_INDEX
            }
        }
    ];

    const completion = await createAzureOpenAICompletion(systemPrompt,
userPrompt, temperature, dataSources) as AzureOpenAIYourDataResponse;
    console.log('Azure OpenAI Add Your Own Data Output: \n',
completion.choices[0]?.message);
    for (let citation of completion.choices[0]?.message?.context?.citations
?? []) {
        console.log('Citation Path:', citation.filepath);
    }
    return completion.choices[0]?.message?.content?.trim() ?? '';
}

```

Notice the following functionality in the `getAzureOpenAIBYODCompletion()` function:

- A `dataSources` property is created which contains the AI Search resource's `key`, `endpoint`, and `index_name` values that were added to the `.env` file earlier in this exercise
- The `createAzureOpenAICompletion()` function is called with the `systemPrompt`, `userPrompt`, `temperature`, and `dataSources` values. This function is used to call Azure OpenAI API and return the results.
- Once the response is returned, the document citations are logged to the console. The completion message content is then returned to the caller.

6. A few final points to consider before moving on to the next exercise:

- The sample application uses a single index in Azure AI Search. You can use multiple indexes and data sources with Azure OpenAI. The `dataSources` property in the `getAzureOpenAIBYODCompletion()` function can be updated to include multiple data sources as needed.
- Security must be carefully evaluated with this type of scenario. Users should't be able to ask questions and get results from documents that they aren't able to access.

7. Now that you've learned about Azure OpenAI, prompts, completions, and how you can use your own data, let's go to the next exercise to learn how communication features can be used to enhance the application. If you'd like to learn more about Azure OpenAI, view the [Get started with Azure OpenAI Service](#) training content. Additional information about using your own data with Azure OpenAI can be found in the [Azure OpenAI on your data](#) documentation.

Next Step

[Communication: Creating an Azure Communication Services Resource](#)

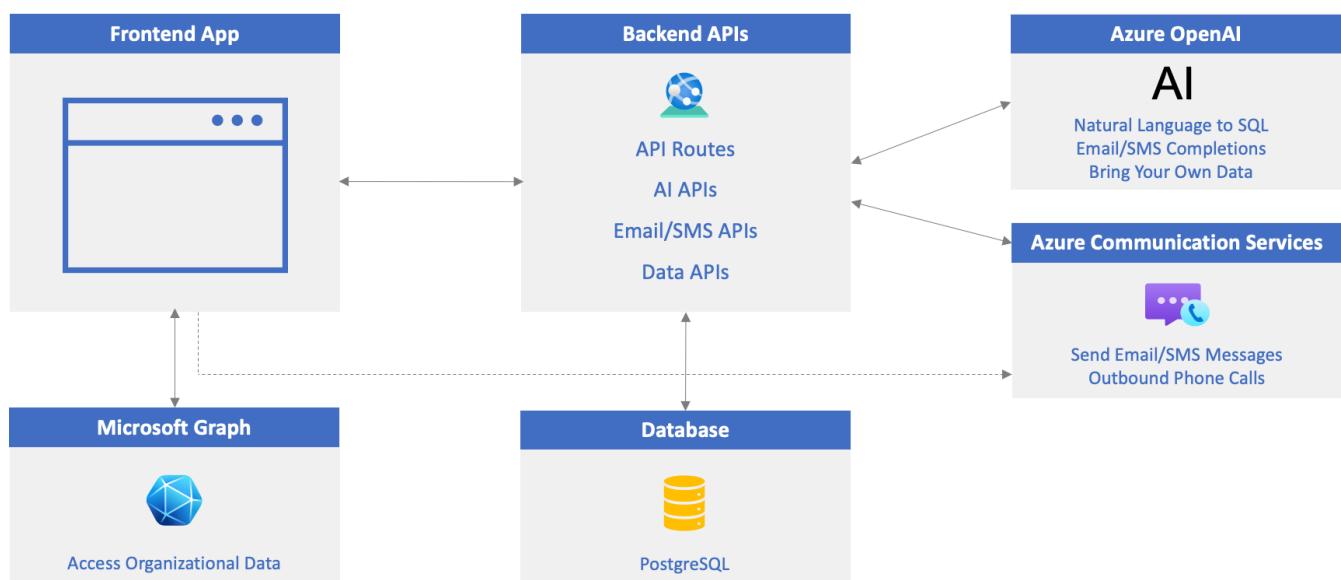
Communication: Creating an Azure Communication Services Resource

06/13/2025

Effective communication is essential for successful custom business applications. By using Azure Communication Services (ACS), you can add features such as phone calls, live chat, audio/video calls, and email and SMS messaging to your applications. Earlier, you learned how Azure OpenAI can generate completions for email and SMS messages. Now, you'll learn how to send the messages. Together, ACS and OpenAI can enhance your applications by simplifying communication, improving interactions, and boosting business productivity.

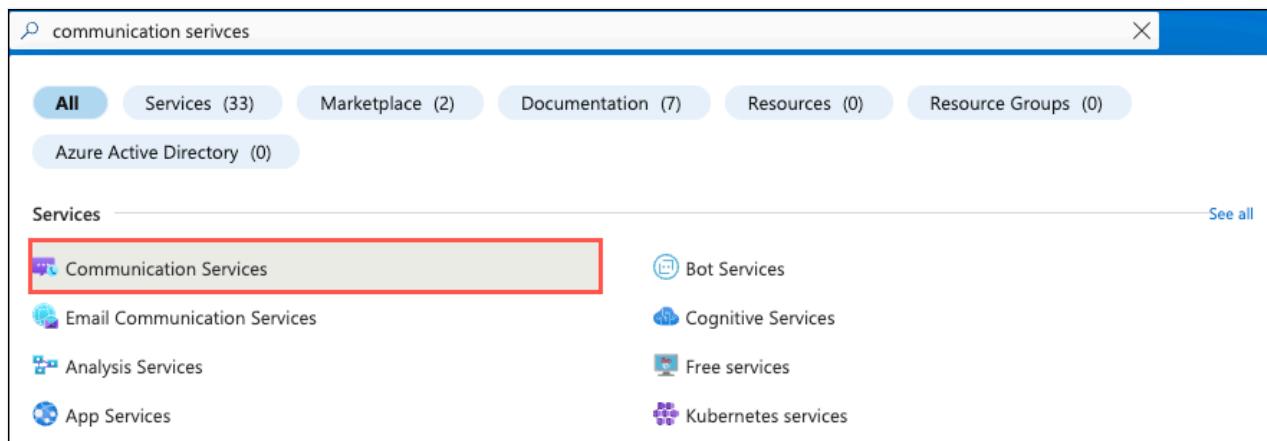
In this exercise, you will:

- Create an Azure Communication Services (ACS) resource.
- Add a toll-free phone number with calling and SMS capabilities.
- Connect an email domain.
- Update the project's `.env` file with values from your ACS resource.



Create an Azure Communication Services Resource

1. Visit the [Azure portal](#) in your browser and sign in if you haven't already.
2. Type *communication services* in the **search bar** at the top of the page and select **Communication Services** from the options that appear.



3. Select **Create** in the toolbar.

4. Perform the following tasks:

- Select your Azure subscription.
- Select the resource group to use (create a new one if one doesn't exist).
- Enter an ACS resource name. It must be a unique value.
- Select a data location.

5. Select **Review + Create** followed by **Create**.

6. You've successfully created a new Azure Communication Services resource! Next, you'll enable phone calling and SMS capabilities. You'll also connect an email domain to the resource.

Enable Phone Calling and SMS Capabilities

1. Add a phone number and ensure that the phone number has calling capabilities enabled. You'll use this phone number to call out to a phone from the app.

- Select **Telephony and SMS --> Phone numbers** from the Resource menu.
- Select **+ Get** in the toolbar (or select the **Get a number** button) and enter the following information:
 - **Country or region:** `United States`
 - **Number type:** `Toll-free`

ⓘ Note

A credit card is required on your Azure subscription to create the toll-free number. If you don't have a card on file, feel free to skip adding a phone number and jump to the next section of the exercise that connects an email

domain. You can still use the app, but won't be able to call out to a phone number.

- **Number:** Select **Add to cart** for one of the phone numbers listed.

2. Select **Next**, review the phone number details, and select **Buy now**.

① Note

SMS verification for toll-free numbers is now mandatory in the United States and Canada. To enable SMS messaging, you must submit verification after the phone number purchase. While this tutorial won't go through that process, you can select **Telephony and SMS --> Regulatory Documents** from the resources menu and add the required validation documentation.

3. Once the phone number is created, select it to get to the **Features** panel. Ensure that the following values are set (they should be set by default):

- In the **Calling** section, select **Make calls**.
- In the **SMS** section, select **Send and receive SMS**.
- Select **Save**.

4. Copy the phone number value into a file for later use. The phone number should follow this general pattern: `+12345678900`.

Connect an Email Domain

1. Perform the following tasks to create a connected email domain for your ACS resource so that you can send email. This will be used to send email from the app.

- Select **Email --> Domains** from the Resource menu.
- Select **Connect domain** from the toolbar.
- Select your **Subscription** and **Resource group**.
- Under the **Email Service** dropdown, select **Add an email service**.
- Give the email service a name such as `acs-demo-email-service`.
- Select **Review + create** followed by **Create**.
- Once the deployment completes, select **Go to resource**, and select **1-click add** to add a free Azure subdomain.
- After the subdomain is added (it'll take a few moments to be deployed), select it.
- Once you're on the **AzureManagedDomain** screen, select **Email services --> MailFrom addresses** from the Resource menu.
- Copy the **MailFrom** value to a file. You'll use it later as you update the `.env` file.

- Go back to your Azure Communication Services resource and select **Email --> Domains** from the resource menu.
- Select **Add domain** and enter the **MailFrom** value from the previous step (ensure you select the correct subscription, resource group, and email service). Select the **Connect** button.

Update the `.env` File

1. Now that your ACS phone number (with calling and SMS enabled) and email domain are ready, update the following keys/values in the `.env` file in your project:

```
ACS_CONNECTION_STRING=<ACS_CONNECTION_STRING>
ACS_PHONE_NUMBER=<ACS_PHONE_NUMBER>
ACS_EMAIL_ADDRESS=<ACS_EMAIL_ADDRESS>
CUSTOMER_EMAIL_ADDRESS=<EMAIL_ADDRESS_TO_SEND_EMAIL_TO>
CUSTOMER_PHONE_NUMBER=<UNITED_STATES_BASED_NUMBER_TO_SEND_SMS_TO>
```

- **ACS_CONNECTION_STRING**: The `connection string` value from the **Keys** section of your ACS resource.
- **ACS_PHONE_NUMBER**: Assign your toll-free number to the `ACS_PHONE_NUMBER` value.
- **ACS_EMAIL_ADDRESS**: Assign your email "MailTo" address value.
- **CUSTOMER_EMAIL_ADDRESS**: Assign an email address you'd like email to be sent to from the app (since the customer data in the app's database is only sample data). You can use a personal email address.
- **CUSTOMER_PHONE_NUMBER**: You'll need to provide a United States based phone number (as of today) due to additional verification that is required in other countries/regions for sending SMS messages. If you don't have a US-based number, you can leave it empty.

Start/Rotate the Application and API Servers

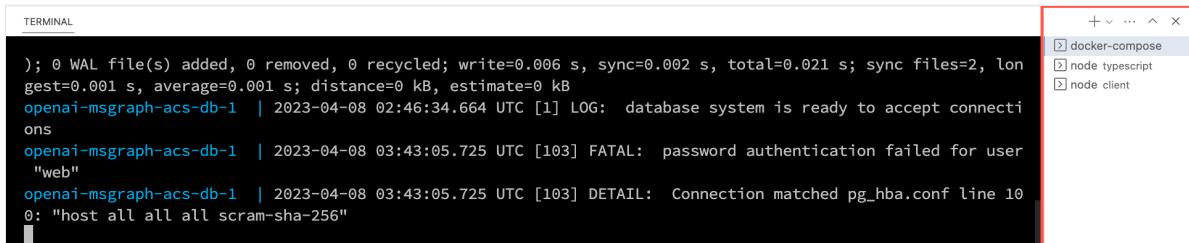
Perform **one** of the following steps based on the exercises you completed up to this point:

- If you started the database, API server, and web server in an earlier exercise, you need to stop the API server and web server and restart them to pick up the `.env` file changes. You can leave the database running.

Locate the terminal windows running the API server and web server and press **CTRL + C** to stop them. Start them again by typing `npm start` in each terminal window and pressing **Enter**. Continue to the next exercise.

- If you haven't started the database, API server, and web server yet, complete the following steps:

1. In the following steps you'll create three terminal windows in Visual Studio Code.



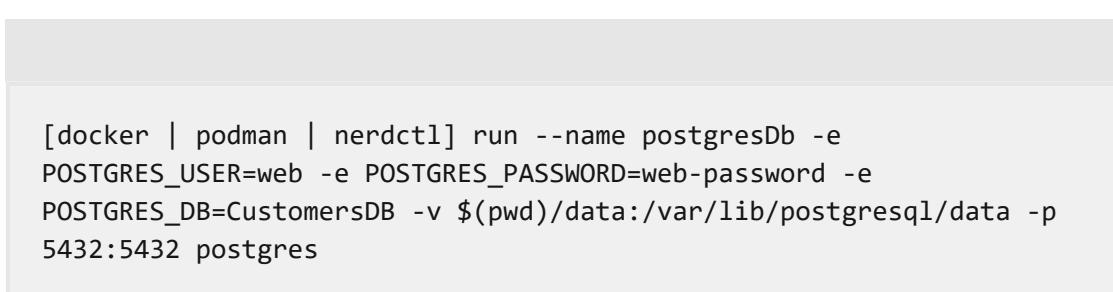
A screenshot of the Visual Studio Code interface. On the left is a terminal window displaying PostgreSQL logs. On the right is a sidebar with three tabs: "docker-compose", "node typescript", and "node client". A red box highlights the sidebar area.

```
TERMINAL
);
0 WAL file(s) added, 0 removed, 0 recycled; write=0.006 s, sync=0.002 s, total=0.021 s; sync files=2, longest=0.001 s, average=0.001 s; distance=0 kB, estimate=0 kB
openai-msgraph-acss-db-1 | 2023-04-08 02:46:34.664 UTC [1] LOG:  database system is ready to accept connections
openai-msgraph-acss-db-1 | 2023-04-08 03:43:05.725 UTC [103] FATAL:  password authentication failed for user
"web"
openai-msgraph-acss-db-1 | 2023-04-08 03:43:05.725 UTC [103] DETAIL:  Connection matched pg_hba.conf line 10
0: "host all all all scram-sha-256"
```

2. Right-click on the `.env` file in the Visual Studio Code file list and select **Open in Integrated Terminal**. Ensure that your terminal is at the root of the project - `openai-acss-msgraph` - before continuing.

3. Choose from **one** of the following options to start the PostgreSQL database:

- If you have [Docker Desktop](#) installed and running, run `docker-compose up` in the terminal window and press **Enter**.
- If you have Podman with [podman-compose](#) installed and running, run `podman-compose up` in the terminal window and press **Enter**.
- To run the PostgreSQL container directly using either Docker Desktop, Podman, nerdctl, or another container runtime you have installed, run the following command in the terminal window:
 - Mac, Linux, or Windows Subsystem for Linux (WSL):



```
[docker | podman | nerdctl] run --name postgresDb -e
POSTGRES_USER=web -e POSTGRES_PASSWORD=web-password -e
POSTGRES_DB=CustomersDB -v $(pwd)/data:/var/lib/postgresql/data -p
5432:5432 postgres
```

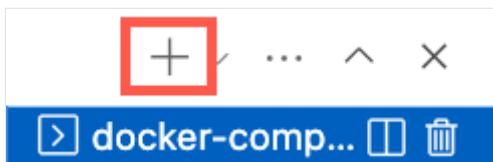
- Windows with PowerShell:



```
[docker | podman] run --name postgresDb -e POSTGRES_USER=web -e
POSTGRES_PASSWORD=web-password -e POSTGRES_DB=CustomersDB -v
```

```
 ${PWD}/data:/var/lib/postgresql/data -p 5432:5432 postgres
```

- Once the database container starts, press the + icon in the Visual Studio Code Terminal toolbar to create a second terminal window.



- cd into the *server/typescript* folder and run the following commands to install the dependencies and start the API server.
 - npm install
 - npm start
- Press the + icon again in the Visual Studio Code Terminal toolbar to create a third terminal window.
- cd into the *client* folder and run the following commands to install the dependencies and start the web server.
 - npm install
 - npm start
- A browser will launch and you'll be taken to <http://localhost:4200>.

A screenshot of a web application titled "Customers App". The page has a dark blue header with the title. Below it is a light blue sidebar with a "Customers" icon and a "Filter Data:" input field. The main content area is a table with the following columns: Company, First Name, Last Name, City, Email, and Phone. There are five rows of data:

Company	First Name	Last Name	City	Email	Phone
Adatum Corporation	Jane	Doe	New York	jane.doe@example.com	+1 (555) 123-4567
Adventure Works Cycles	John	Smith	London	john.smith@example.com	+1 (555) 123-7654
Contoso Pharmaceuticals	Peter	Gibbons	Austin	peter.gibbons@example.com	+1 (555) 321-1234
Tailwind Traders	Lisa	Taylor	Sydney	lisa.taylor@example.com	+1 (555) 123-1234

Each row has a "Contact Customer" button. At the bottom left, there is a text input field containing a custom query: "Get the total revenue for all orders. Group by company and include the city." Below the input field are two buttons: "Run Query" and "Reset Data".

Next Step

Communication: Making a Phone Call

Communication: Making a Phone Call

06/13/2025

Integrating Azure Communication Services' phone calling capabilities into a custom Line of Business (LOB) application offers several key benefits to businesses and their users:

- Enables seamless and real-time communication between employees, customers, and partners, directly from within the LOB application, eliminating the need to switch between multiple platforms or devices.
- Enhances the user experience and improves overall operational efficiency.
- Facilitates rapid problem resolution, as users can quickly connect with relevant support teams or subject matter experts quickly and easily.

In this exercise, you will:

- Explore the phone calling feature in the application.
- Walk through the code to learn how the phone calling feature is built.

Using the Phone Calling Feature

1. In the [previous exercise](#) you created an Azure Communication Services (ACS) resource and started the database, web server, and API server. You also updated the following values in the `.env` file.

```
ACS_CONNECTION_STRING=<ACS_CONNECTION_STRING>
ACS_PHONE_NUMBER=<ACS_PHONE_NUMBER>
ACS_EMAIL_ADDRESS=<ACS_EMAIL_ADDRESS>
CUSTOMER_EMAIL_ADDRESS=<EMAIL_ADDRESS_TO_SEND_EMAIL_TO>
CUSTOMER_PHONE_NUMBER=<UNITED_STATES_BASED_NUMBER_TO_SEND_SMS_TO>
```

Ensure you've completed the [previous exercise](#) before continuing.

2. Go back to the browser (`http://localhost:4200`), locate the datagrid, and select **Contact Customer** followed by **Call Customer** in the first row.

Company	First Name	Last Name	City	Email	Phone		
Adatum Corporation	Jane	Doe	New York	jane.doe@example.com	+1 (555) 123-4567	View Related Content	Contact Customer
Adventure Works Cycles	John	Smith	London	john.smith@example.com	+1 (555) 123-7654	View Related Content	Call Customer
Contoso Pharmaceuticals	Peter	Gibbons	Austin	peter.gibbons@example.com	+1 (555) 321-1234	View Related Content	Email/SMS Customer
Tailwind Traders	Lisa	Taylor	Sydney	lisa.taylor@example.com	+1 (555) 123-1234	View Related Content	Contact Customer

3. A phone call component will be added into the header. Enter your phone number you'd like to call (ensure it starts with + and includes the country code) and select **Call**. You will be prompted to allow access to your microphone.



4. Select **Hang Up** to end the call. Select **Close** to close the phone call component.

Exploring the Phone Calling Code

💡 Tip

If you're using Visual Studio Code, you can open files directly by selecting:

- Windows/Linux: `Ctrl + P`
- Mac: `Cmd + P`

Then type the name of the file you want to open.

1. Open the `customers-list.component.ts` file. The full path to the file is `client/src/app/customers-list/customers-list.component.ts`.
2. Note that `openCallDialog()` sends a `CustomerCall` message and the customer phone number using an event bus.

TypeScript

```
openCallDialog(data: Phone) {  
    this.eventBus.emit({ name: Events.CustomerCall, value: data });  
}
```

❗ Note

The event bus code can be found in the `eventbus.service.ts` file if you're interested in exploring it more. The full path to the file is `client/src/app/core/eventbus.service.ts`.

3. The header component's `ngOnInit()` function subscribes to the `CustomerCall` event sent by the event bus and displays the phone call component. You can find this code in `header.component.ts`.

TypeScript

```

ngOnInit() {
    this.subscription.add(
        this.eventBus.on(Events.CustomerCall, (data: Phone) => {
            this.callVisible = true; // Show phone call component
            this.callData = data; // Set phone number to call
        })
    );
}

```

4. Open *phone-call.component.ts*. Take a moment to explore the code. The full path to the file is *client/src/app/phone-call/phone-call.component.ts*. Note the following key features:

- Retrieves an Azure Communication Services access token by calling the `acsService.getAcsToken()` function in `ngOnInit()`;
- Adds a "phone dialer" to the page. You can see the dialer by clicking on the phone number input in the header.
- Starts and ends a call using the `startCall()` and `endCall()` functions respectively.

5. Before looking at the code that makes the phone call, let's examine how the ACS access token is retrieved and how phone calling objects are created. Locate the `ngOnInit()` function in *phone-call.component.ts*.

TypeScript

```

async ngOnInit() {
    if (ACS_CONNECTION_STRING) {
        this.subscription.add(
            this.acsService.getAcsToken().subscribe(async (user: AcsUser) =>
{
            const callClient = new CallClient();
            const tokenCredential = new
AzureCommunicationTokenCredential(user.token);
            this.callAgent = await
callClient.createCallAgent(tokenCredential);
        })
    );
}
}

```

This function performs the following actions:

- Retrieves an ACS userId and access token by calling the `acsService.getAcsToken()` function.
- Once the access token is retrieved, the code performs the following actions:
 - Creates a new instance of `CallClient` and `AzureCommunicationTokenCredential` using the access token.

- Creates a new instance of `CallAgent` using the `callClient` and `AzureCommunicationTokenCredential` objects. Later you'll see that `CallAgent` is used to start and end a call.

6. Open `acs.services.ts` and locate the `getAcsToken()` function. The full path to the file is `client/src/app/core/acs.service.ts`. The function makes an HTTP GET request to the `/acstoken` route exposed by the API server.

TypeScript

```
getAcsToken(): Observable<AcsUser> {
    return this.http.get<AcsUser>(this.apiUrl + 'acstoken')
        .pipe(
            catchError(this.handleError)
        );
}
```

7. An API server function named `createACSToken()` retrieves the `userId` and access token and returns it to the client. It can be found in the `server/typescript/acs.ts` file.

TypeScript

```
import { CommunicationIdentityClient } from '@azure/communication-identity';

const connectionString = process.env.ACS_CONNECTION_STRING as string;

async function createACSToken() {
    if (!connectionString) return { userId: '', token: '' };

    const tokenClient = new CommunicationIdentityClient(connectionString);
    const { user, token } = await tokenClient.createUserAndToken(['voip']);
    return { userId: user.communicationUserId, token };
}
```

This function performs the following actions:

- Checks if an ACS `connectionString` value is available. If not, returns an object with an empty `userId` and `token`.
- Creates a new instance of `CommunicationIdentityClient` and passes the `connectionString` value to it.
- Creates a new user and token using `tokenClient.createUserAndToken()` with the "voip" scope.
- Returns an object containing the `userId` and `token` values.

8. Now that you've seen how the userId and token are retrieved, go back to `phone-call.component.ts` and locate the `startCall()` function.

9. This function is called when **Call** is selected in the phone call component. It uses the `callAgent` object mentioned earlier to start a call. The `callAgent.startCall()` function accepts an object representing the number to call and the ACS phone number used to make the call.

TypeScript

```
startCall() {
    this.call = this.callAgent?.startCall(
        [{ phoneNumber: this.customerPhoneNumber }], {
            alternateCallerId: { phoneNumber: this.fromNumber }
        });
    console.log('Calling: ', this.customerPhoneNumber);
    console.log('Call id: ', this.call?.id);
    this.inCall = true;

    // Adding event handlers to monitor call state
    this.call?.on('stateChanged', () => {
        console.log('Call state changed: ', this.call?.state);
        if (this.call?.state === 'Disconnected') {
            console.log('Call ended. Reason: ', this.call.callEndReason);
            this.inCall = false;
        }
    });
}
```

10. The `endCall()` function is called when **Hang Up** is selected in the phone call component.

TypeScript

```
endCall() {
    if (this.call) {
        this.call.hangUp({ forEveryone: true });
        this.call = undefined;
        this.inCall = false;
    }
    else {
        this.hangup.emit();
    }
}
```

If a call is in progress, the `call.hangUp()` function is called to end the call. If no call is in progress, the `hangup` event is emitted to the header parent component to hide the phone call component.

11. Before moving on to the next exercise, let's review the key concepts covered in this exercise:

- An ACS userId and access token are retrieved from the API server using the `acsService.createUserAndToken()` function.
- The token is used to create a `CallClient` and `CallAgent` object.
- The `CallAgent` object is used to start and end a call by calling the `callAgent.startCall()` and `callAgent.hangUp()` functions respectively.

12. Now that you've learned how phone calling can be integrated into an application, let's switch our focus to using Azure Communication Services to send email and SMS messages.

Next Step

[Communication: Sending Email and SMS Messages](#)

Communication: Sending Email and SMS Messages

06/13/2025

In addition to phone calls, Azure Communication Services can also send email and SMS messages. This can be useful when you want to send a message to a customer or other user directly from the application.

In this exercise, you will:

- Explore how email and SMS messages can be sent from the application.
- Walk through the code to learn how the email and SMS functionality is implemented.

Using the Email and SMS Features

1. In a [previous exercise](#) you created an Azure Communication Services (ACS) resource and started the database, web server, and API server. You also updated the following values in the `.env` file.

```
ACS_CONNECTION_STRING=<ACS_CONNECTION_STRING>
ACS_PHONE_NUMBER=<ACS_PHONE_NUMBER>
ACS_EMAIL_ADDRESS=<ACS_EMAIL_ADDRESS>
CUSTOMER_EMAIL_ADDRESS=<EMAIL_ADDRESS_TO_SEND_EMAIL_TO>
CUSTOMER_PHONE_NUMBER=<UNITED_STATES_BASED_NUMBER_TO_SEND_SMS_TO>
```

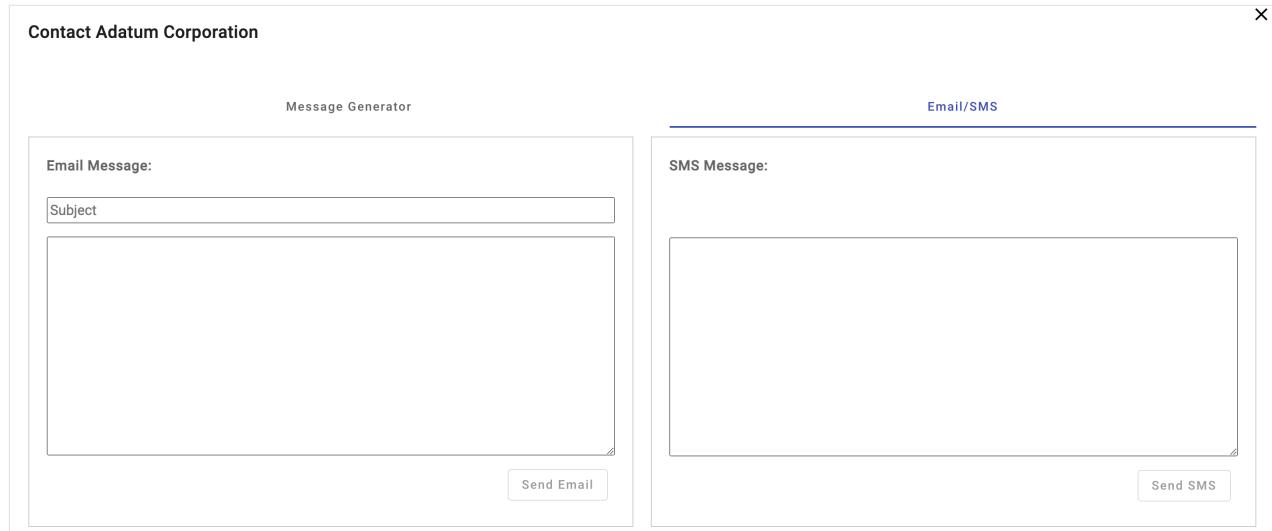
Ensure you've completed the [previous exercise](#) before continuing.

2. Go back to the browser (`http://localhost:4200`) and select **Contact Customer** followed by **Email/SMS Customer** in the first row.

The screenshot shows a web application titled "Customers App". The main page displays a table of customer data with columns: Company, First Name, Last Name, City, Email, and Phone. Below the table is a "Contact Customer" button. A context menu is open over the first row of the table, listing options: "View Related Content", "Contact Customer", "Call Customer", and "Email/SMS Customer". The "Email/SMS Customer" option is highlighted with a blue background and a white font. A mouse cursor is pointing at this highlighted option.

3. Select the **Email/SMS** tab and perform the following tasks:

- Enter an Email Subject and Body and select the Send Email button.
- Enter an SMS message and select the Send SMS button.



ⓘ Note

SMS verification for toll-free numbers is now mandatory in the United States and Canada. To enable SMS messaging, you must submit verification after the phone number purchase. While this tutorial won't go through that process, you can select **Telephony and SMS --> Regulatory Documents** from your Azure Communication Services resource in the Azure portal and add the required validation documentation.

4. Check that you received the email and SMS messages. SMS functionality will only work if you submitted the regulatory documents mentioned in the previous note. As a reminder, the email message will be sent to the value defined for `CUSTOMER_EMAIL_ADDRESS` and the SMS message will be sent to the value defined for `CUSTOMER_PHONE_NUMBER` in the `.env` file. If you weren't able to supply a United States based phone number to use for SMS messages you can skip that step.

ⓘ Note

If you don't see the email message in your inbox for the address you defined for `CUSTOMER_EMAIL_ADDRESS` in the `.env` file, check your spam folder.

Exploring the Email Code

💡 Tip

If you're using Visual Studio Code, you can open files directly by selecting:

- Windows/Linux: `Ctrl + P`
- Mac: `Cmd + P`

Then type the name of the file you want to open.

1. Open the `customers-list.component.ts` file. The full path to the file is `client/src/app/customers-list/customers-list.component.ts`.
2. When you selected **Contact Customer** followed by **Email/SMS Customer** in the datagrid, the `customer-list` component displayed a dialog box. This is handled by the `openEmailSmsDialog()` function in the `customer-list.component.ts` file.

```
TypeScript

openEmailSmsDialog(data: any) {
    if (data.phone && data.email) {
        // Create the data for the dialog
        let dialogData: EmailSmsDialogData = {
            prompt: '',
            title: `Contact ${data.company}`,
            company: data.company,
            customerName: data.first_name + ' ' + data.last_name,
            customerEmailAddress: data.email,
            customerPhoneNumber: data.phone
        }

        // Open the dialog
        const dialogRef = this.dialog.open(EmailSmsDialogComponent, {
            data: dialogData
        });

        // Subscribe to the dialog afterClosed observable to get the dialog
        result
            this.subscription.add(
                dialogRef.afterClosed().subscribe((response: EmailSmsDialogData)
            => {
                console.log('SMS dialog result:', response);
                if (response) {
                    dialogData = response;
                }
            })
        );
    }
    else {
        alert('No phone number available.');
    }
}
```

The `openEmailSmsDialog()` function performs the following tasks:

- Checks to see if the `data` object (which represents the row from the datagrid) contains a `phone` and `email` property. If it does, it creates a `dialogData` object that contains the information to pass to the dialog.
- Opens the `EmailSmsDialogComponent` dialog box and passes the `dialogData` object to it.
- Subscribes to the `afterClosed()` event of the dialog box. This event is fired when the dialog box is closed. The `response` object contains the information that was entered into the dialog box.

3. Open the `email-sms-dialog.component.ts` file. The full path to the file is `client/src/app/email-sms-dialog/email-sms-dialog.component.ts`.

4. Locate the `sendEmail()` function:

```
TypeScript

sendEmail() {
    if (this.featureFlags.acsEmailEnabled) {
        // Using CUSTOMER_EMAIL_ADDRESS instead of this.data.email for
        testing purposes
        this.subscription.add(
            this.acsService.sendEmail(this.emailSubject, this.emailBody,
                this.getFirstName(this.data.customerName),
                CUSTOMER_EMAIL_ADDRESS /* this.data.email */)
            .subscribe(res => {
                console.log('Email sent:', res);
                if (res.status) {
                    this.emailSent = true;
                }
            })
        );
    } else {
        this.emailSent = true; // Used when ACS email isn't enabled
    }
}
```

The `sendEmail()` function performs the following tasks:

- Checks to see if the `acsEmailEnabled` feature flag is set to `true`. This flag checks to see if the `ACS_EMAIL_ADDRESS` environment variable has an assigned value.
- If `acsEmailEnabled` is true, the `acsService.sendEmail()` function is called and the email subject, body, customer name, and customer email address are passed. Because the database contains sample data, the `CUSTOMER_EMAIL_ADDRESS` environment variable is used instead of `this.data.email`. In a real-world application the `this.data.email` value would be used.

- Subscribes to the `sendEmail()` function in the `acsService` service. This function returns an RxJS observable that contains the response from the client-side service.
- If the email was sent successfully, the `emailSent` property is set to `true`.

5. To provide better code encapsulation and reuse, client-side services such as `acs.service.ts` are used throughout the application. This allows all ACS functionality to be consolidated into a single place.
6. Open `acs.service.ts` and locate the `sendEmail()` function. The full path to the file is `client/src/app/core/acs.service.ts`.

TypeScript

```
sendEmail(subject: string, message: string, customerName: string,
customerEmailAddress: string) : Observable<EmailSmsResponse> {
    return this.http.post<EmailSmsResponse>(this.apiUrl + 'sendEmail', {
        subject, message, customerName, customerEmailAddress })
    .pipe(
        catchError(this.handleError)
    );
}
```

The `sendEmail()` function in `AcsService` performs the following tasks:

- Calls the `http.post()` function and passes the email subject, message, customer name, and customer email address to it. The `http.post()` function returns an RxJS observable that contains the response from the API call.
- Handles any errors returned by the `http.post()` function using the RxJS `catchError` operator.

7. Now let's examine how the application interacts with the ACS email feature. Open the `acs.ts` file and locate the `sendEmail()` function. The full path to the file is `server/typescript/acs.ts`.

8. The `sendEmail()` function performs the following tasks:

- Creates a new `EmailClient` object and passes the ACS connection string to it (this value is retrieved from the `ACS_CONNECTION_STRING` environment variable).

TypeScript

```
const emailClient = new EmailClient(connectionString);
```

- Creates a new `EmailMessage` object and passes the sender, subject, message, and recipient information.

TypeScript

```
const msgObject: EmailMessage = {
  senderAddress: process.env.ACS_EMAIL_ADDRESS as string,
  content: {
    subject: subject,
    plainText: message,
  },
  recipients: {
    to: [
      {
        address: customerEmailAddress,
        displayName: customerName,
      },
    ],
  },
};
```

- Sends the email using the `emailClient.beginSend()` function and returns the response. Although the function is only sending to one recipient in this example, the `beginSend()` function can be used to send to multiple recipients as well.

TypeScript

```
const poller = await emailClient.beginSend(msgObject);
```

- Waits for the `poller` object to signal it's done and sends the response to the caller.

Exploring the SMS Code

- Go back to the `email-sms-dialog.component.ts` file that you opened earlier. The full path to the file is `client/src/app/email-sms-dialog/email-sms-dialog.component.ts`.
- Locate the `sendSms()` function:

TypeScript

```
sendSms() {
  if (this.featureFlags.acsPhoneEnabled) {
    // Using CUSTOMER_PHONE_NUMBER instead of
    this.data.customerPhoneNumber for testing purposes
    this.subscription.add(
      this.acsService.sendSms(this.smsMessage, CUSTOMER_PHONE_NUMBER /* *
      this.data.customerPhoneNumber */)
```

```

        .subscribe(res => {
            if (res.status) {
                this.smsSent = true;
            }
        })
    );
}
else {
    this.smsSent = true;
}
}

```

The `sendSMS()` function performs the following tasks:

- Checks to see if the `acsPhoneEnabled` feature flag is set to `true`. This flag checks to see if the `ACS_PHONE_NUMBER` environment variable has an assigned value.
- If `acsPhoneEnabled` is true, the `acsService.SendSms()` function is called and the SMS message and customer phone number are passed. Because the database contains sample data, the `CUSTOMER_PHONE_NUMBER` environment variable is used instead of `this.data.customerPhoneNumber`. In a real-world application the `this.data.customerPhoneNumber` value would be used.
- Subscribes to the `sendSms()` function in the `acsService` service. This function returns an RxJS observable that contains the response from the client-side service.
- If the SMS message was sent successfully, it sets the `smsSent` property to `true`.

3. Open `acs.service.ts` and locate the `sendSms()` function. The full path to the file is `client/src/app/core/acs.service.ts`.

TypeScript

```

sendSms(message: string, customerPhoneNumber: string) :
Observable<EmailSmsResponse> {
    return this.http.post<EmailSmsResponse>(this.apiUrl + 'sendSms', {
        message, customerPhoneNumber })
    .pipe(
        catchError(this.handleError)
    );
}

```

The `sendSms()` function performs the following tasks:

- Calls the `http.post()` function and passes the message and customer phone number to it. The `http.post()` function returns an RxJS observable that contains the response from the API call.

- Handles any errors returned by the `http.post()` function using the RxJS `catchError` operator.

4. Finally, let's examine how the application interacts with the ACS SMS feature. Open the `acs.ts` file. The full path to the file is `server/typescript/acs.ts` and locate the `sendSms()` function.

5. The `sendSms()` function performs the following tasks:

- Creates a new `SmsClient` object and passes the ACS connection string to it (this value is retrieved from the `ACS_CONNECTION_STRING` environment variable).

TypeScript

```
const smsClient = new SmsClient(connectionString);
```

- Calls the `smsClient.send()` function and passes the ACS phone number (`from`), customer phone number (`to`), and SMS message:

TypeScript

```
const sendResults = await smsClient.send({
  from: process.env.ACS_PHONE_NUMBER as string,
  to: [customerPhoneNumber],
  message: message
});
return sendResults;
```

- Returns the response to the caller.

6. You can learn more about ACS email and SMS functionality in the following articles:

- [Email in Azure Communication Services](#)
- [SMS in Azure Communication Services](#)

7. Before moving on to the next exercise, let's review the key concepts covered in this exercise:

- The `acs.service.ts` file encapsulates the ACS email and SMS functionality used by the client-side application. It handles the API calls to the server and returns the response to the caller.
- The server-side API uses the ACS `EmailClient` and `SmsClient` objects to send email and SMS messages.

8. Now that you've learned how email and SMS messages can be sent, let's switch our focus to integrating organizational data into the application.

Next Step

Organizational Data: Creating a Microsoft Entra ID App Registration

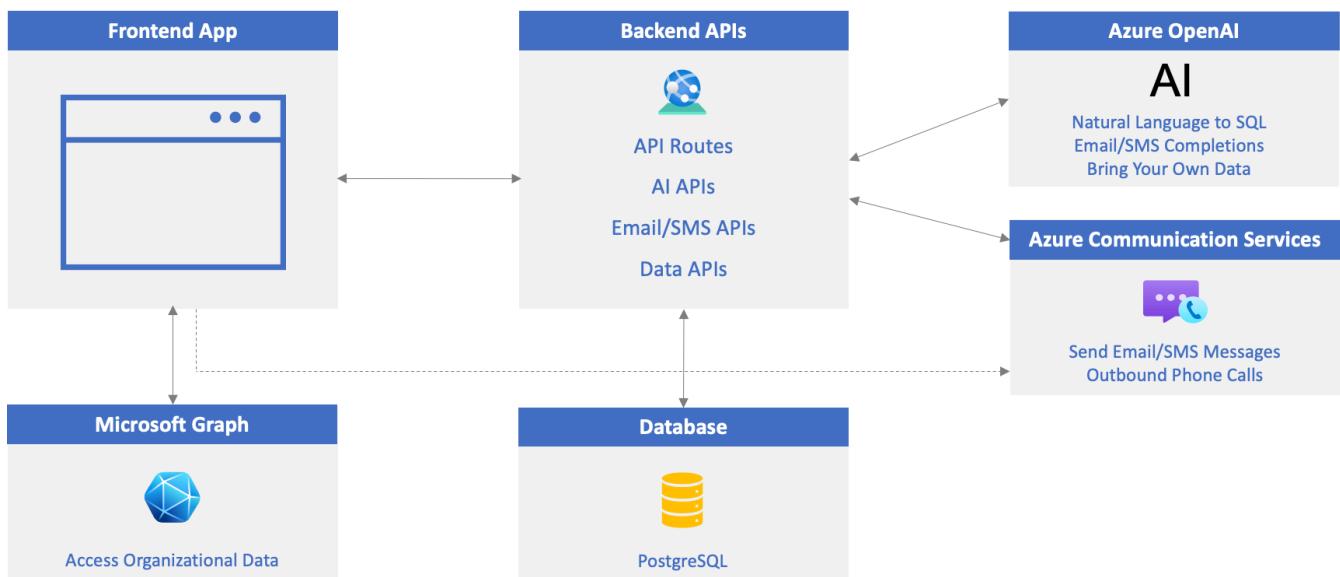
Organizational Data: Creating a Microsoft Entra ID App Registration

06/13/2025

Enhance user productivity by integrating organizational data (emails, files, chats, and calendar events) directly into your custom applications. By using Microsoft Graph APIs and Microsoft Entra ID, you can seamlessly retrieve and display relevant data within your apps, reducing the need for users to switch context. Whether it's referencing an email sent to a customer, reviewing a Teams message, or accessing a file, users can quickly find the information they need without leaving your app, streamlining their decision-making process.

In this exercise, you will:

- Create a Microsoft Entra ID app registration so that Microsoft Graph can access organizational data and bring it into the app.
- Locate `team` and `channel` IDs from Microsoft Teams that are needed to send chat messages to a specific channel.
- Update the project's `.env` file with values from your Microsoft Entra ID app registration.



Create a Microsoft Entra ID App Registration

1. Go to [Azure portal](#) and select Microsoft Entra ID.
2. Select **Manage** --> **App registrations** followed by **+ New registration**.
3. Fill in the new app registration form details as shown below and select **Register**:
 - Name: *microsoft-graph-app*

- Supported account types: *Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant)*
- Redirect URI:
 - Select **Single-page application (SPA)** and enter `http://localhost:4200` in the **Redirect URI** field.
- Select **Register** to create the app registration.

Register an application

The user-facing display name for this application (this can be changed later).

microsoft-graph-app



Supported account types

Who can use this application or access this API?

- Accounts in this organizational directory only (Microsoft only - Single tenant)
- Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant)
- Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- Personal Microsoft accounts only

[Help me choose...](#)

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Single-page application (SPA)

`http://localhost:4200`



[By proceeding, you agree to the Microsoft Platform Policies](#)

Register

1. Select **Overview** in the resource menu and copy the **Application (client) ID** value to your clipboard.

The screenshot shows the Azure portal's "Overview" page for the "microsoft-graph-app" application. The left sidebar has "Overview" selected. The main area shows the application's details under the "Essentials" section. The "Application (client) ID" field is highlighted with a red box. Other fields shown include "Display name" (microsoft-graph-app), "Object ID" (redacted), "Directory (tenant) ID" (redacted), "Supported account types" (Multiple organizations), and links for "Client credentials", "Add a certificate or secret", "Redirect URLs", "Application ID URI", and "Managed application in local directory".

Update the Project's .env File

1. Open the .env file in your editor and assign the Application (client) ID value to ENTRAIID_CLIENT_ID.

```
ENTRAID_CLIENT_ID=<APPLICATION_CLIENT_ID_VALUE>
```

2. If you'd like to enable the ability to send a message from the app into a Teams Channel, sign in to [Microsoft Teams](#) using your Microsoft 365 dev tenant account (this is mentioned in the pre-reqs for the tutorial).
3. Once you're signed in, expand a team, and find a channel that you want to send messages to from the app. For example, you might select the Company team and the General channel (or whatever team/channel you'd like to use).



4. In the team header, click on the three dots (...) and select Get link to team.
5. In the link that appears in the popup window, the team ID is the string of letters and numbers after `team/`. For example, in the link "<https://teams.microsoft.com/l/team/19%3ae9b9.../>", the team ID is `19%3ae9b9...` up to the following `/` character.
6. Copy the team ID and assign it to `TEAM_ID` in the .env file.
7. In the channel header, click on the three dots (...) and select Get link to channel.
8. In the link that appears in the popup window, the channel ID is the string of letters and numbers after `channel/`. For example, in the link "<https://teams.microsoft.com/l/channel/19%3aQK02.../>", the channel ID is `19%3aQK02...` up to the following `/` character.
9. Copy the channel ID and assign it to `CHANNEL_ID` in the .env file.
10. Save the `env` file before continuing.

Start/Rotate the Application and API Servers

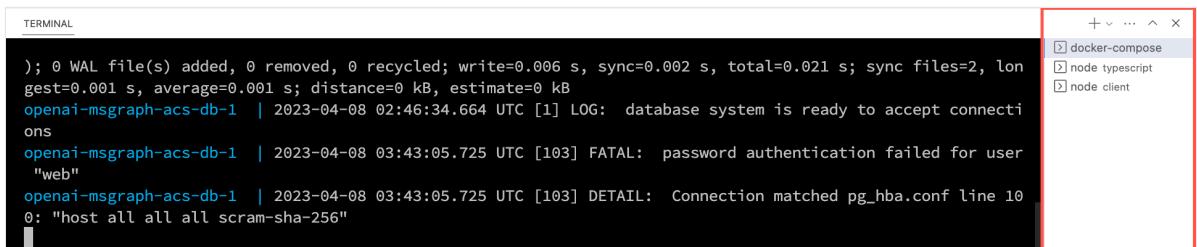
Perform **one** of the following steps based on the exercises you completed up to this point:

- If you started the database, API server, and web server in an earlier exercise, you need to stop the API server and web server and restart them to pick up the `.env` file changes. You can leave the database running.

Locate the terminal windows running the API server and web server and press `CTRL + C` to stop them. Start them again by typing `npm start` in each terminal window and pressing `Enter`. Continue to the next exercise.

- If you haven't started the database, API server, and web server yet, complete the following steps:

1. In the following steps you'll create three terminal windows in Visual Studio Code.



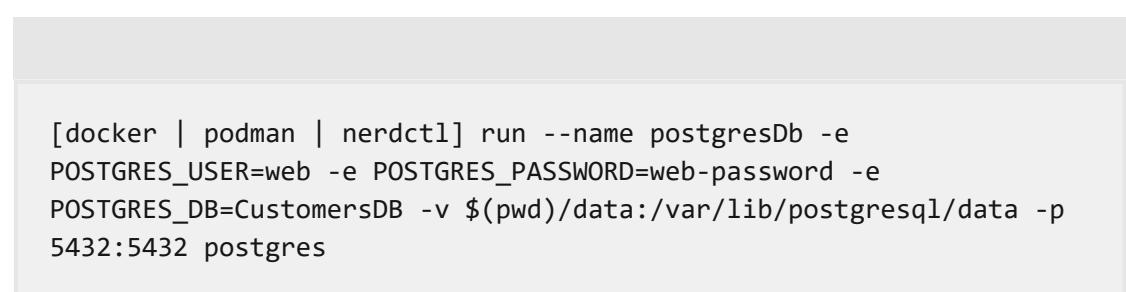
The screenshot shows the Visual Studio Code interface. On the left is the 'TERMINAL' tab. The terminal window displays log messages from a PostgreSQL database container named 'openai-msgraph-acs-db-1'. The logs show the database system ready to accept connections, a password authentication failure for user 'web', and a connection matched pg_hba.conf line 10. The right side of the interface shows a sidebar with project files: 'docker-compose', 'node typescript', and 'node client'. A red box highlights the terminal window and the sidebar area.

```
( ); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.006 s, sync=0.002 s, total=0.021 s; sync files=2, longest=0.001 s, average=0.001 s; distance=0 kB, estimate=0 kB
openai-msgraph-acs-db-1 | 2023-04-08 02:46:34.664 UTC [1] LOG:  database system is ready to accept connections
openai-msgraph-acs-db-1 | 2023-04-08 03:43:05.725 UTC [103] FATAL:  password authentication failed for user "web"
openai-msgraph-acs-db-1 | 2023-04-08 03:43:05.725 UTC [103] DETAIL:  Connection matched pg_hba.conf line 10
0: "host all all all scram-sha-256"
```

2. Right-click on the `.env` file in the Visual Studio Code file list and select **Open in Integrated Terminal**. Ensure that your terminal is at the root of the project - `openai-acs-msgraph` - before continuing.

3. Choose from **one** of the following options to start the PostgreSQL database:

- If you have [Docker Desktop](#) installed and running, run `docker-compose up` in the terminal window and press `Enter`.
- If you have Podman with [podman-compose](#) installed and running, run `podman compose up` in the terminal window and press `Enter`.
- To run the PostgreSQL container directly using either Docker Desktop, Podman, nerdctl, or another container runtime you have installed, run the following command in the terminal window:
- Mac, Linux, or Windows Subsystem for Linux (WSL):



The screenshot shows a terminal window with a single command to run a PostgreSQL container. The command uses '[docker | podman | nerdctl]' to run a container named 'postgresDb' with environment variables: POSTGRES_USER=web, POSTGRES_PASSWORD=web-password, and POSTGRES_DB=CustomersDB. It also mounts the current directory as '/data' and binds port 5432 to the container's port 5432.

```
[docker | podman | nerdctl] run --name postgresDb -e
POSTGRES_USER=web -e POSTGRES_PASSWORD=web-password -e
POSTGRES_DB=CustomersDB -v $(pwd)/data:/var/lib/postgresql/data -p
5432:5432 postgres
```

- o Windows with PowerShell:

```
[docker | podman] run --name postgresDb -e POSTGRES_USER=web -e POSTGRES_PASSWORD=web-password -e POSTGRES_DB=CustomersDB -v ${PWD}/data:/var/lib/postgresql/data -p 5432:5432 postgres
```

4. Once the database container starts, press the + icon in the Visual Studio Code **Terminal toolbar** to create a second terminal window.



5. `cd` into the *server/typescript* folder and run the following commands to install the dependencies and start the API server.

- o `npm install`
- o `npm start`

6. Press the + icon again in the Visual Studio Code **Terminal toolbar** to create a third terminal window.

7. `cd` into the *client* folder and run the following commands to install the dependencies and start the web server.

- o `npm install`
- o `npm start`

8. A browser will launch and you'll be taken to <http://localhost:4200>.

Company	First Name	Last Name	City	Email	Phone	
Adatum Corporation	Jane	Doe	New York	jane.doe@example.com	+1 (555) 123-4567	
Adventure Works Cycles	John	Smith	London	john.smith@example.com	+1 (555) 123-7654	
Contoso Pharmaceuticals	Peter	Gibbons	Austin	peter.gibbons@example.com	+1 (555) 321-1234	
Tailwind Traders	Lisa	Taylor	Sydney	lisa.taylor@example.com	+1 (555) 123-1234	

Custom Query:
Get the total revenue for all orders. Group by company and include the city.

Run Query Reset Data

Next Step

Organizational Data: Signing In a User and Getting an Access Token

Organizational Data: Signing In a User and Getting an Access Token

06/13/2025

Users need to authenticate with Microsoft Entra ID in order for Microsoft Graph to access organizational data. In this exercise, you'll see how the Microsoft Graph Toolkit's `mgt-login` component can be used to authenticate users and retrieve an access token. The access token can then be used to make calls to Microsoft Graph.

ⓘ Note

If you're new to Microsoft Graph, you can learn more about it in the [Microsoft Graph Fundamentals](#) learning path.

In this exercise, you will:

- Learn how to associate a Microsoft Entra ID app with the Microsoft Graph Toolkit to authenticate users and retrieve organizational data.
- Learn about the importance of scopes.
- Learn how the Microsoft Graph Toolkit's *mgt-login* component can be used to authenticate users and retrieve an access token.

Using the Sign In Feature

1. In the [previous exercise](#), you created an app registration in Microsoft Entra ID and started the application server and API server. You also updated the following values in the `.env` file (`TEAM_ID` and `CHANNEL_ID` are optional):

```
ENTRAID_CLIENT_ID=<APPLICATION_CLIENT_ID_VALUE>
TEAM_ID=<TEAMS_TEAM_ID>
CHANNEL_ID=<TEAMS_CHANNEL_ID>
```

Ensure you've completed the [previous exercise](#) before continuing.

2. Go back to the browser (`http://localhost:4200`), select **Sign In** in the header, and sign in using an admin user account from your Microsoft 365 Developer tenant.

💡 Tip

Sign in with your Microsoft 365 developer tenant admin account. You can view other users in your developer tenant by going to the [Microsoft 365 admin center](#).

3. If you're signing in to the application for the first time, you'll be prompted to consent to the permissions requested by the application. You'll learn more about these permissions (also called "scopes") in the next section as you explore the code. Select **Accept** to continue.
4. Once you're signed in, you should see the name of the user displayed in the header.



Exploring the Sign In Code

Now that you've signed in, let's look at the code used to sign in the user and retrieve an access token and user profile. You'll learn about the *mgt-login* web component that's part of the Microsoft Graph Toolkit.

💡 Tip

If you're using Visual Studio Code, you can open files directly by selecting:

- Windows/Linux: `Ctrl + P`
- Mac: `Cmd + P`

Then type the name of the file you want to open.

1. Open *client/package.json* and notice that the `@microsoft/mgt` and `@microsoft/mgt-components` packages are included in the dependencies. The `@microsoft/mgt` package contains MSAL (Microsoft Authentication Library) provider features and web components such as *mgt-login* and others that can be used to sign in users and retrieve and display organizational data.
2. Open *client/src/main.ts* and notice the following imports from the `@microsoft/mgt-components` package. The imported symbols are used to register the Microsoft Graph Toolkit components that are used in the application.

TypeScript

```
import { registerMgt LoginComponent, registerMgtSearchResultsComponent,
registerMgtPersonComponent, } from '@microsoft/mgt-components';
```

3. Scroll to the bottom of the file and note the following code:

```
TypeScript
```

```
registerMgtLoginComponent();
registerMgtSearchResultsComponent();
registerMgtPersonComponent();
```

This code registers the `mgt-login`, `mgt-search-results`, and `mgt-person` web components and enables them for use in the application.

4. To use the `mgt-login` component to sign in users, the Microsoft Entra ID app's client Id (stored in the `.env` file as `ENTRAID_CLIENT_ID`) needs to be referenced and used.
5. Open `graph.service.ts` and locate the `init()` function. The full path to the file is `client/src/app/core/graph.service.ts`. You'll see the following import and code:

```
TypeScript
```

```
import { Msal2Provider, Providers, ProviderState } from '@microsoft/mgt';

init() {
    if (!this.featureFlags.microsoft365Enabled) return;

    if (!Providers.globalProvider) {
        console.log('Initializing Microsoft Graph global provider...');
        Providers.globalProvider = new Msal2Provider({
            clientId: ENTRAID_CLIENT_ID,
            scopes: ['User.Read', 'Presence.Read', 'Chat.ReadWrite',
            'Calendars.Read',
                'ChannelMessage.Read.All', 'ChannelMessage.Send',
            'Files.Read.All', 'Mail.Read']
        });
    }
    else {
        console.log('Global provider already initialized');
    }
}
```

This code creates a new `Msal2Provider` object, passing the Microsoft Entra ID client Id from your app registration and the `scopes` for which the app will request access. The `scopes` are used to request access to Microsoft Graph resources that the app will access. After the `Msal2Provider` object is created, it's assigned to the `Providers.globalProvider` object, which is used by Microsoft Graph Toolkit components to retrieve data from Microsoft Graph.

6. Open `header.component.html` in your editor and locate the `mgt-login` component. The full path to the file is `client/src/app/header/header.component.html`.

HTML

```
@if (this.featureFlags.microsoft365Enabled) {  
  <mgt-login class="mgt-dark" (loginCompleted)="loginCompleted()"></mgt-  
  login>  
}
```

The `mgt-login` component enables user sign in and provides access to a token that is used with Microsoft Graph. Upon successful sign in, the `loginCompleted` event is triggered, which calls the `loginCompleted()` function. Although `mgt-login` is used within an Angular component in this example, it is compatible with any web application.

Display of the `mgt-login` component depends on the `featureFlags.microsoft365Enabled` value being set to `true`. This custom flag checks for the presence of the `ENTRAID_CLIENT_ID` environment variable to confirm that the application is properly configured and able to authenticate against Microsoft Entra ID. The flag is added to accommodate cases where users choose to complete only the AI or Communication exercises within the tutorial, rather than following the entire sequence of exercises.

7. Open `header.component.ts` and locate the `loginCompleted` function. This function is called when the `loginCompleted` event is emitted and handles retrieving the signed in user's profile using `Providers.globalProvider`.

TypeScript

```
async loginCompleted() {  
  const me = await Providers.globalProvider.graph.client.api('me').get();  
  this.userLoggedIn.emit(me);  
}
```

In this example, a call is being made to the Microsoft Graph `me` API to retrieve the user's profile (`me` represents the current signed in user). The `this.userLoggedIn.emit(me)` code statement emits an event from the component to pass the profile data to the parent component. The parent component is `app.component.ts` file in this case, which is the root component for the application.

To learn more about the `mgt-login` component, visit the [Microsoft Graph Toolkit](#) documentation.

8. Now that you've logged into the application, let's look at how organizational data can be retrieved.

Next Step

Organizational Data: Retrieving Files, Chats, and Sending Messages to Teams

Organizational Data: Retrieving Files, Chats, and Sending Messages to Teams

06/13/2025

In today's digital environment, users work with a wide array of organizational data, including emails, chats, files, calendar events, and more. This can lead to frequent context shifts—switching between tasks or applications—which can disrupt focus and reduce productivity. For example, a user working on a project might need to switch from their current application to Outlook to find crucial details in an email or switch to OneDrive for Business to find a related file. This back-and-forth action disrupts focus and wastes time that could be better spent on the task at hand.

To enhance efficiency, you can integrate organizational data directly into the applications users use everyday. By bringing in organizational data to your applications, users can access and manage information more seamlessly, minimizing context shifts and improving productivity. Additionally, this integration provides valuable insights and context, enabling users to make informed decisions and work more effectively.

In this exercise, you will:

- Learn how the *mgt-search-results* web component in the Microsoft Graph Toolkit can be used to search for files.
- Learn how to call Microsoft Graph directly to retrieve files from OneDrive for Business and chat messages from Microsoft Teams.
- Understand how to send chat messages to Microsoft Teams channels using Microsoft Graph.

Using the Organizational Data Feature

1. In a [previous exercise](#) you created an app registration in Microsoft Entra ID and started the application server and API server. You also updated the following values in the `.env` file.

```
ENTRAID_CLIENT_ID=<APPLICATION_CLIENT_ID_VALUE>
TEAM_ID=<TEAMS_TEAM_ID>
CHANNEL_ID=<TEAMS_CHANNEL_ID>
```

Ensure you've completed the [previous exercise](#) before continuing.

2. Go back to the browser (<http://localhost:4200>). If you haven't already signed in, select **Sign In** in the header, and sign in with a user from your Microsoft 365 Developer tenant.

 **Note**

In addition to authenticating the user, the *mgt-login* web component also retrieves an access token that can be used by Microsoft Graph to access files, chats, emails, calendar events, and other organizational data. The access token contains the scopes (permissions) such as `Chat.ReadWrite`, `Files.Read.All`, and others that you saw earlier:

TypeScript

```
Providers.globalProvider = new Msal2Provider({
  clientId: ENTRAID_CLIENT_ID, // retrieved from .env file
  scopes: ['User.Read', 'Presence.Read', 'Chat.ReadWrite',
  'Calendars.Read',
  'ChannelMessage.Read.All', 'ChannelMessage.Send',
  'Files.Read.All', 'Mail.Read']
});
```

3. Select **View Related Content** for the *Adatum Corporation* row in the datagrid. This will cause organizational data such as files, chats, emails, and calendar events to be retrieved using Microsoft Graph. Once the data loads, it'll be displayed below the datagrid in a tabbed interface. It's important to mention that you may not see any data at this point since you haven't added any files, chats, emails, or calendar events for the user in your Microsoft 365 developer tenant yet. Let's fix that in the next step.

Adatum Corporation



Files



Adatum-Corporation-PO-2005.docx

Modified 12/29/2022



Adatum-Corporation-Logo.svg

Modified 1/2/2023

4. Your Microsoft 365 tenant may not have any related organizational data for *Adatum Corporation* at this stage. To add some sample data, perform at least one of the following actions:

- Add files by visiting <https://onedrive.com> and signing in using your Microsoft 365 Developer tenant credentials.
 - Select **My files** in the left navigation.
 - Select **+ Add new** and then **Folder upload** from the menu.
 - Select the *openai-acs-msgraph/customer documents* folder from the project you cloned.



OneDrive



Add new



Folder



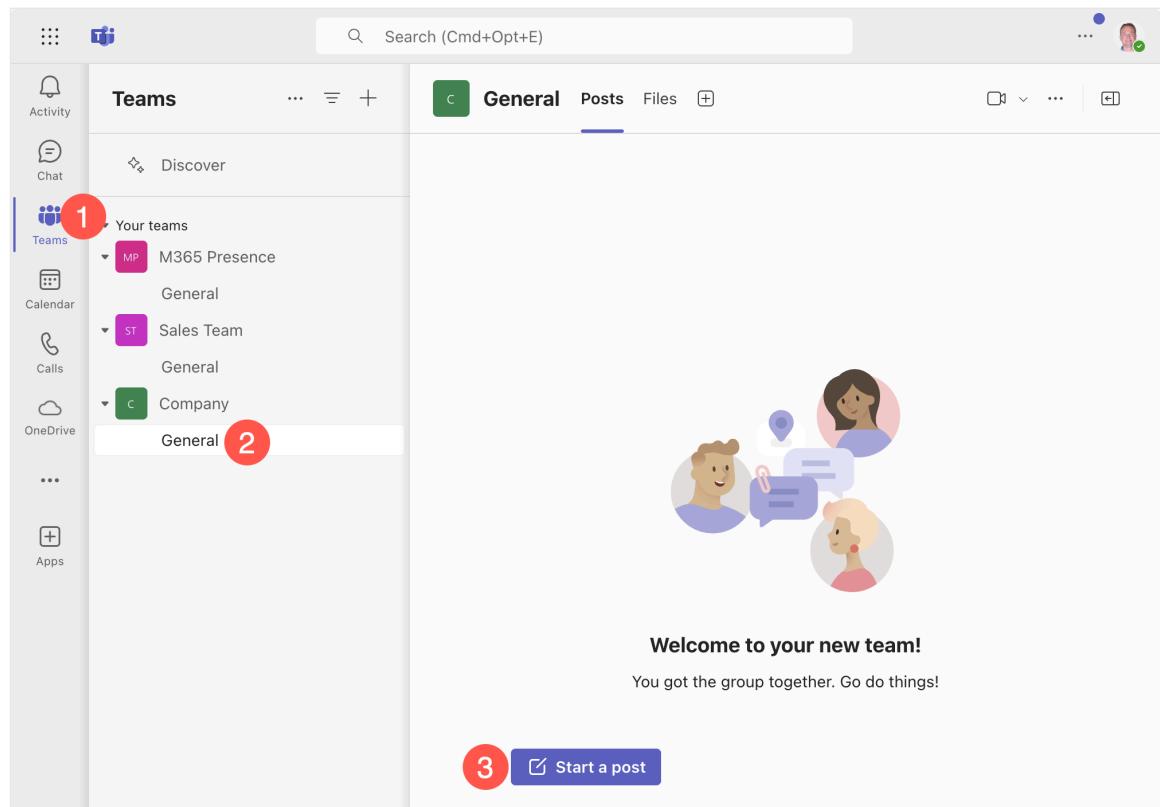
Files upload



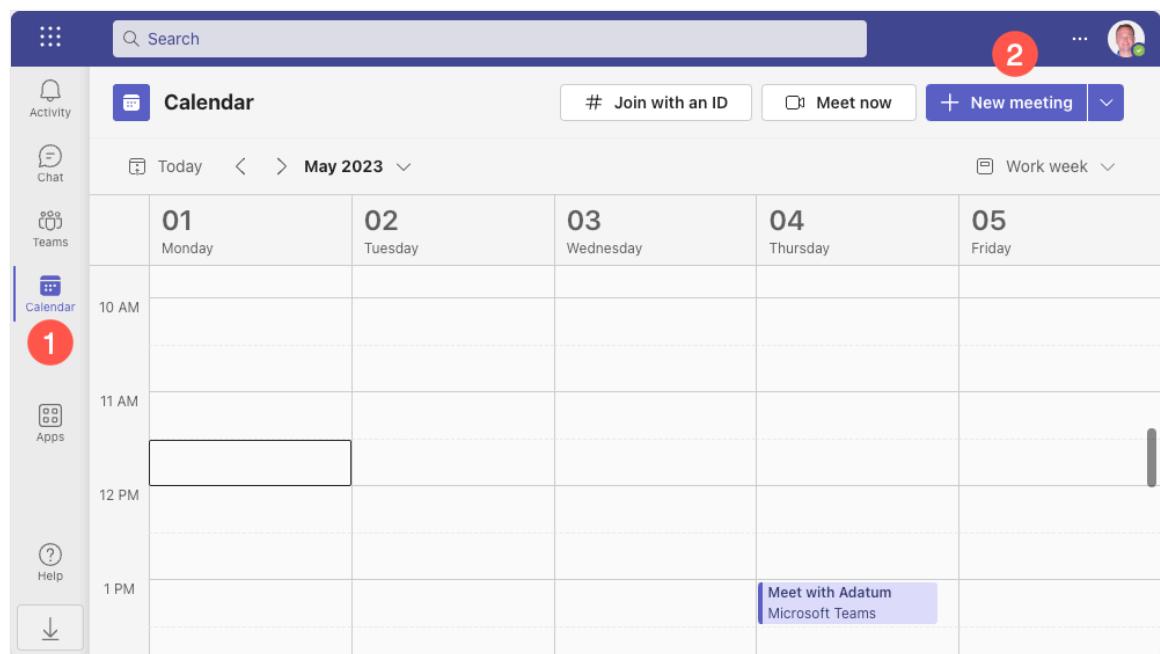
Folder upload

- Add chat messages and calendar events by visiting <https://teams.microsoft.com> and signing in using your Microsoft 365 Developer tenant credentials.
 - Select **Teams** in the left navigation.
 - Select a team and channel.
 - Select **Start a post**.
 - Enter *New order placed for Adatum Corporation* for the subject and any additional text you'd like to add in the message body. Select the **Post** button.

Feel free to add additional chat messages that mention other companies used in the application such as *Adventure Works Cycles*, *Contoso Pharmaceuticals*, and *Tailwind Traders*.

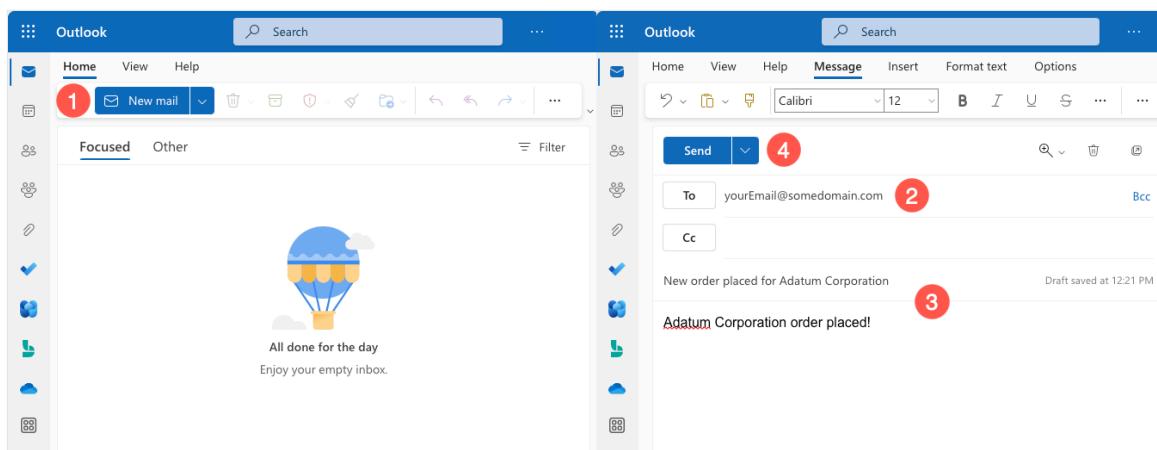


- Select **Calendar** in the left navigation.
- Select **New meeting**.
- Enter "Meet with Adatum Corporation about project schedule" for the title and body.
- Select **Save**.



- Add emails by visiting <https://outlook.com> and signing in using your Microsoft 365 Developer tenant credentials.

- Select **New mail**.
- Enter your personal email address in the **To** field.
- Enter *New order placed for Adatum Corporation* for the subject and anything you'd like for the body.
- Select **Send**.



5. Go back to the application in the browser and refresh the page. Select **View Related Content** again for the *Adatum Corporation* row. You should now see data displayed in the tabs depending upon which tasks you performed in the previous step.
6. Let's explore the code that enables the organizational data feature in the application. To retrieve the data, the client-side portion of the application uses the access token retrieved by the *mgt-login* component you looked at earlier to make calls to Microsoft Graph APIs.

Exploring Files Search Code

Tip

If you're using Visual Studio Code, you can open files directly by selecting:

- Windows/Linux: `Ctrl + P`
- Mac: `Cmd + P`

Then type the name of the file you want to open.

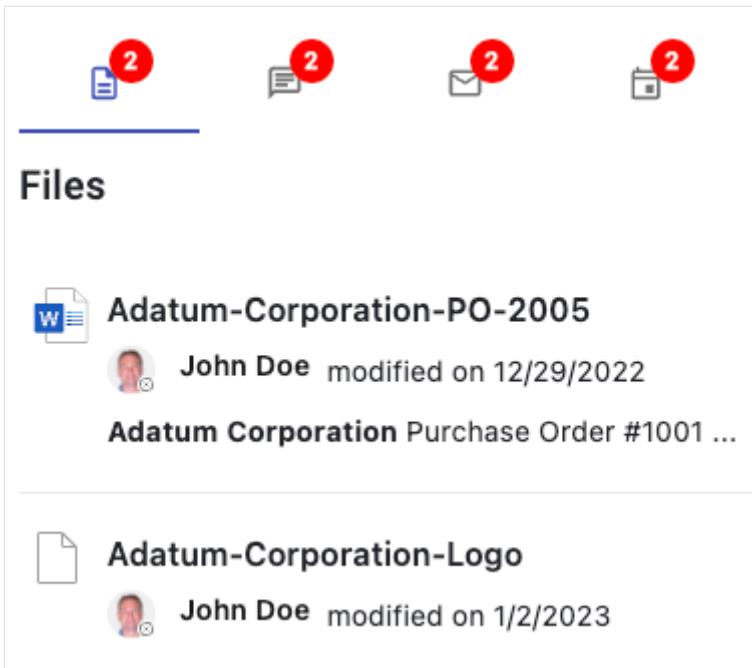
1. Let's start by looking at how file data is retrieved from OneDrive for Business. Open *files.component.html* and take a moment to look through the code. The full path to the file is *client/src/app/files/files.component.html*.
2. Locate the *mgt-search-results* component and note the following attributes:

HTML

```
<mgt-search-results
  class="search-results"
  entity-types="driveItem"
  [queryString]="searchText"
  (dataChange)="dataChange($any($event))"
/>
```

The *mgt-search-results* component is part of the Microsoft Graph Toolkit and as the name implies, it's used to display search results from Microsoft Graph. The component uses the following features in this example:

- The `class` attribute is used to specify that the `search-results` CSS class should be applied to the component.
- The `entity-types` attribute is used to specify the type of data to search for. In this case, the value is `driveItem` which is used to search for files in OneDrive for Business.
- The `queryString` attribute is used to specify the search term. In this case, the value is bound to the `searchText` property which is passed to the *files* component when the user selects **View Related Content** for a row in the datagrid. The square brackets around `queryString` indicate that the property is bound to the `searchText` value.
- The `dataChange` event fires when the search results change. In this case, a customer function named `dataChange()` is called in the *files* component and the event data is passed to the function. The parenthesis around `dataChange` indicate that the event is bound to the `dataChange()` function.
- Since no custom template is supplied, the default template built into `mgt-search-results` is used to display the search results.



3. An alternative to using components such as `mgt-search-results`, is to call Microsoft Graph APIs directly using code. To see how that works, open the `graph.service.ts` file and locate the `searchFiles()` function. The full path to the file is `client/src/app/core/graph.service.ts`.

- You'll notice that a `query` parameter is passed to the function. This is the search term that's passed as the user selects **View Related Content** for a row in the datagrid. If no search term is passed, an empty array is returned.

TypeScript

```
async searchFiles(query: string) {
    const files: DriveItem[] = [];
    if (!query) return files;

    ...
}
```

- A filter is then created that defines the type of search to perform. In this case the code is searching for files in OneDrive for Business so `driveItem` is used just as you saw earlier with the `mgt-search-results` component. This is the same as passing `driveItem` to `entity-types` in the `mgt-search-results` component that you saw earlier. The `query` parameter is then added to the `queryString` filter along with `ContentType:Document`.

TypeScript

```
const filter = {
    "requests": [
        {
            "entityTypes": [
```

```

        "driveItem"
    ],
    "query": {
        "queryString": `${query} AND ContentType:Document`
    }
}
];
}
;
```

- A call is then made to the `/search/query` Microsoft Graph API using the `Providers.globalProvider.graph.client.api()` function. The `filter` object is passed to the `post()` function which sends the data to the API.

TypeScript

```

const searchResults = await
Providers.globalProvider.graph.client.api('/search/query').post(filter);
```

- The search results are then iterated through to locate `hits`. Each `hit` contains information about a document that was found. A property named `resource` contains the document metadata and is added to the `files` array.

TypeScript

```

if (searchResults.value.length !== 0) {
    for (const hitContainer of searchResults.value[0].hitsContainers) {
        if (hitContainer.hits) {
            for (const hit of hitContainer.hits) {
                files.push(hit.resource);
            }
        }
    }
}
```

- The `files` array is then returned to the caller.

TypeScript

```

return files;
```

4. Looking through this code you can see that the *mgt-search-results* web component you explored earlier does a lot of work for you and significantly reduces the amount of code you have to write! However, there may be scenarios where you want to call Microsoft Graph directly to have more control over the data that's sent to the API or how the results are processed.

5. Open the `files.component.ts` file and locate the `search()` function. The full path to the file is `client/src/app/files/files.component.ts`.

Although the body of this function is commented out due to the `mgt-search-results` component being used, the function could be used to make the call to Microsoft Graph when the user selects **View Related Content** for a row in the datagrid. The `search()` function calls `searchFiles()` in `graph.service.ts` and passes the `query` parameter to it (the name of the company in this example). The results of the search are then assigned to the `data` property of the component.

TypeScript

```
override async search(query: string) {
    this.data = await this.graphService.searchFiles(query);
}
```

The files component can then use the `data` property to display the search results. You could handle this using custom HTML bindings or by using another Microsoft Graph Toolkit control named `mgt-file-list`. Here's an example of binding the `data` property to one of the component's properties named `files` and handling the `itemClick` event as the user interacts with a file.

HTML

```
<mgt-file-list (itemClick)="itemClick($any($event))" [files]="data"></mgt-
file-list>
```

6. Whether you choose to use the `mgt-search-results` component shown earlier or write custom code to call Microsoft Graph will depend on your specific scenario. In this example, the `mgt-search-results` component is used to simplify the code and reduce the amount of work you have to do.

Exploring Teams Chat Messages Search Code

1. Go back to `graph.service.ts` and locate the `searchChatMessages()` function. You'll see that it's similar to the `searchFiles()` function you looked at previously.

- It posts filter data to Microsoft Graph's `/search/query` API and converts the results into an array of objects that have information about the `teamId`, `channelId`, and `messageId` that match the search term.
- To retrieve the Teams channel messages, a second call is made to the `/teams/${chat.teamId}/channels/${chat.channelId}/messages/${chat.messageId}` API

and the `teamId`, `channelId`, and `messageId` are passed. This returns the full message details.

- Additional filtering tasks are performed and the resulting messages are returned from `searchChatMessages()` to the caller.

2. Open the `chats.component.ts` file and locate the `search()` function. The full path to the file is `client/src/app/chats/chats.component.ts`. The `search()` function calls `searchChatMessages()` in `graph.service.ts` and passes the `query` parameter to it.

TypeScript

```
override async search(query: string) {
    this.data = await this.graphService.searchChatMessages(query);
}
```

The results of the search are assigned to the `data` property of the component and data binding is used to iterate through the results array and render the data. This example uses an *Angular Material card* component to display the search results.

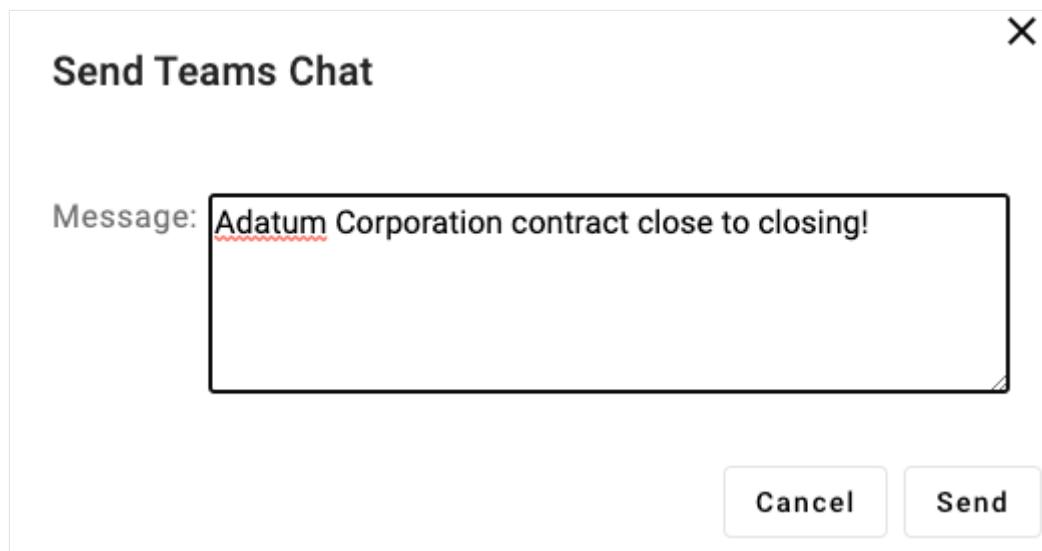
HTML

```
@if (this.data.length) {
    <div>
        @for (chatMessage of this.data;track chatMessage.id) {
            <mat-card>
                <mat-card-header>
                    <mat-card-title [innerHTML]="chatMessage.summary"></mat-
card-title>
                    <!-- <mat-card-subtitle [innerHTML]="chatMessage.body">-->
                </mat-card-header>
                <mat-card-actions>
                    <a mat-stroked-button color="basic"
[ href ]="chatMessage.webUrl" target="_blank">View Message</a>
                </mat-card-actions>
            </mat-card>
        }
    </div>
}
```

The screenshot shows the Microsoft Teams Chats page. At the top, there are four icons with red notification bubbles: a document (2), a message (2), an email (2), and a calendar (2). Below these, the word "Chats" is displayed. A button labeled "New Teams Chat" is visible. Two messages from a channel named "Adatum Corporation test2" are listed. The first message is "Adatum Corporation orders information: Order#: 1001 Total: \$5000 Quantity: 100" with a "View Message" button. The second message is partially visible below it.

Sending a Message to a Microsoft Teams Channel

1. In addition to searching for Microsoft Teams chat messages, the application also allows a user to send messages to a Microsoft Teams channel. This can be done by calling the `/teams/${teamId}/channels/${channelId}/messages` endpoint of Microsoft Graph.



2. In the following code you'll see that a URL is created that includes the `teamId` and `channelId` values. Environment variable values are used for the team ID and channel ID in this example but those values could be dynamically retrieved as well using Microsoft Graph. The `body` constant contains the message to send. A POST request is then made and the results are returned to the caller.

TypeScript

```
async sendTeamsChat(message: string): Promise<TeamsDialogData> {
    if (!message) new Error('No message to send.');
    if (!TEAM_ID || !CHANNEL_ID) new Error('Team ID or Channel ID not set in')
```

```
environment variables. Please set TEAM_ID and CHANNEL_ID in the .env file.');

  const url =
`https://graph.microsoft.com/v1.0/teams/${TEAM_ID}/channels/${CHANNEL_ID}/mes-
sages`;
  const body = {
    "body": {
      "contentType": "html",
      "content": message
    }
  };
  const response = await
Providers.globalProvider.graph.client.api(url).post(body);
  return {
    id: response.id,
    teamId: response.channelIdentity.teamId,
    channelId: response.channelIdentity.channelId,
    message: response.body.content,
    webUrl: response.webUrl,
    title: 'Send Teams Chat'
  };
}
```

3. Leveraging this type of functionality in Microsoft Graph provides a great way to enhance user productivity by allowing users to interact with Microsoft Teams directly from the application they're already using.

Next Step

Organizational Data: Retrieving Emails and Calendar Events

Organizational Data: Retrieving Emails and Calendar Events

06/13/2025

In the previous exercise you learned how to retrieve files from OneDrive for Business and chats from Microsoft Teams using Microsoft Graph and the *mgt-search-results* component from Microsoft Graph Toolkit. You also learned how to send messages to Microsoft Teams channels. In this exercise, you'll learn how to retrieve email messages and calendar events from Microsoft Graph and integrate them into the application.

In this exercise, you will:

- Learn how the *mgt-search-results* web component in the Microsoft Graph Toolkit can be used to search for emails and calendar events.
- Learn how to customize the *mgt-search-results* component to render search results in a custom way.
- Learn how to call Microsoft Graph directly to retrieve emails and calendar events.

Exploring Email Messages Search Code

💡 Tip

If you're using Visual Studio Code, you can open files directly by selecting:

- Windows/Linux: `Ctrl + P`
- Mac: `Cmd + P`

Then type the name of the file you want to open.

1. In a [previous exercise](#) you created an app registration in Microsoft Entra ID and started the application server and API server. You also updated the following values in the `.env` file.

```
ENTRAID_CLIENT_ID=<APPLICATION_CLIENT_ID_VALUE>
TEAM_ID=<TEAMS_TEAM_ID>
CHANNEL_ID=<TEAMS_CHANNEL_ID>
```

Ensure you've completed the [previous exercise](#) before continuing.

2. Open `emails.component.html` and take a moment to look through the code. The full path to the file is `client/src/app/emails/emails.component.html`.

3. Locate the `mgt-search-results` component:

HTML

```
<mgt-search-results
  class="search-results"
  entity-types="message"
  [queryString]="searchText"
  (dataChange)="dataChange($any($event))">
  <template data-type="result-message"></template>
</mgt-search-results>
```

This example of the `mgt-search-results` component is configured the same way as the one you looked at previously. The only difference is that the `entity-types` attribute is set to `message` which is used to search for email messages and an empty template is supplied.

- The `class` attribute is used to specify that the `search-results` CSS class should be applied to the component.
- The `entity-types` attribute is used to specify the type of data to search for. In this case, the value is `message`.
- The `queryString` attribute is used to specify the search term.
- The `dataChange` event fires when the search results change. The emails component's `dataChange()` function is called, the results are passed to it, and a property named `data` is updated in the component.
- An empty template is defined for the component. This type of template is normally used to define how the search results will be rendered. However, in this scenario we're telling the component not to render any message data. Instead, we'll render the data ourselves using standard data binding (Angular is used in this case, but you can use any library/framework you want).

4. Look below the `mgt-search-results` component in `emails.component.html` to find the data bindings used to render the email messages. This example iterates through the `data` property and writes out the email subject, body preview, and a link to view the full email message.

HTML

```
@if (this.data.length) {
  <div>
    @for (email of this.data;track $index) {
      <mat-card>
        <mat-card-header>
```

```

        <mat-card-title>{{email.resource.subject}}</mat-card-title>
        <mat-card-subtitle [innerHTML]="email.resource.bodyPreview">
    </mat-card-subtitle>
        </mat-card-header>
        <mat-card-actions>
            <a mat-stroked-button color="basic"
                [href]="email.resource.webLink" target="_blank">View Email Message</a>
            </mat-card-actions>
        </mat-card>
    }
</div>
}

```

Email

Meet with Adatum Corporation about project schedule
Meet with Adatum Corporation about project schedule
[View Email Message](#)

New order placed for Adatum Corporation
Please review the latest order.
[View Email Message](#)

5. In addition to using the `mgt-search-results` component to retrieve messages, Microsoft Graph provides several APIs that can be used to search emails as well. The `/search/query` API that you saw earlier could certainly be used, but a more straightforward option is the `messages` API.
6. To see how to call this API, go back to `graph.service.ts` and locate the `searchEmailMessages()` function. It creates a URL that can be used to call the `messages` endpoint of Microsoft Graph and assigns the `query` value to the `$search` parameter. The code then makes a GET request and returns the results to the caller. The `$search` operator searches for the `query` value in the subject, body, and sender fields automatically.

TypeScript

```

async searchEmailMessages(query:string) {
    if (!query) return [];

```

```
// The $search operator will search the subject, body, and sender fields automatically
const url = `https://graph.microsoft.com/v1.0/me/messages?
$search="${query}"&$select=subject,bodyPreview,from,toRecipients,receivedDateTime,webLink`;
const response = await Providers.globalProvider.graph.client.api(url).get();
return response.value;
}
```

7. The *emails* component located in *emails.component.ts* calls `searchEmailMessages()` and displays the results in the UI.

TypeScript

```
override async search(query: string) {
    this.data = await this.graphService.searchEmailMessages(query);
}
```

Exploring Calendar Events Search Code

1. Searching for calendar events can also be accomplished using the *mgt-search-results* component. It can handle rendering the results for you, but you can also define your own template which you'll see later in this exercise.
2. Open *calendar-events.component.html* and take a moment to look through the code. The full path to the file is *client/src/app/calendar-events/calendar-events.component.html*. You'll see that it's very similar to the files and emails components you looked at previously.

HTML

```
<mgt-search-results
    class="search-results"
    entity-types="event"
    [queryString]="searchText"
    (dataChange)="dataChange($any($event))">
    <template data-type="result-event"></template>
</mgt-search-results>
```

This example of the *mgt-search-results* component is configured the same way as the ones you looked at previously. The only difference is that the `entity-types` attribute is set to `event` which is used to search for calendar events and an empty template is supplied.

- The `class` attribute is used to specify that the `search-results` CSS class should be applied to the component.
- The `entity-types` attribute is used to specify the type of data to search for. In this case, the value is `event`.
- The `queryString` attribute is used to specify the search term.
- The `dataChange` event fires when the search results change. The calendar `event` component's `dataChange()` function is called, the results are passed to it, and a property named `data` is updated in the component.
- An empty template is defined for the component. In this scenario we're telling the component not to render any data. Instead, we'll render the data ourselves using standard data binding.

3. Immediately below the `mgt-search-results` component in `calendar-events.component.html` you'll find the data bindings used to render the calendar events. This example iterates through the `data` property and writes out the start date, time, and subject of the event. Custom functions included in the component such as `dayFromDateTime()` and `timeRangeFromEvent()` are called to format data properly. The HTML bindings also include a link to view the calendar event in Outlook and the location of the event if one is specified.

HTML

```
@if (this.data.length) {
  <div>
    @for (event of this.data;track $index) {
      <div class="root">
        <div class="time-container">
          <div class="date">{{ dayFromDateTime(event.resource.start.dateTime)}}
          <div class="time">{{ timeRangeFromEvent(event.resource) }}</div>
        </div>
        <div class="separator">
          <div class="vertical-line top"></div>
          <div class="circle">
            @if (!this.event.resource.bodyPreview?.includes('Join Microsoft Teams Meeting')) {
              <div class="inner-circle"></div>
            }
          </div>
          <div class="vertical-line bottom"></div>
        </div>
        <div class="details">
          <div class="subject">{{ event.resource.subject }}</div>
          @if (this.event.resource.location?.displayName) {
            <div class="location">{{ event.resource.location.displayName }}</div>
          }
        </div>
      </div>
    }
  </div>
}
```

```

        <div class="location">
            at
            <a href="https://bing.com/maps/default.aspx?
where1={{event.resource.location.displayName}}"
                target="_blank" rel="noopener"><b>{{
event.resource.location.displayName }}</b></a>
        </div>
    }
    @if (this.event.resource.attendees?.length) {
        <div class="attendees">
            @for (attendee of
this.event.resource.attendees; track attendee.emailAddress.name) {
                <span class="attendee">
                    <mgt-person person-query="
{{attendee.emailAddress.name}}"></mgt-person>
                </span>
            }
        </div>
    }
    @if (this.event.resource.bodyPreview?.includes('Join
Microsoft Teams Meeting')) {
        <div class="online-meeting">
            
                <a class="online-meeting-link" href="{{
event.resource.onlineMeetingUrl }}">
                    Join Teams Meeting
                </a>
            </div>
        }
    </div>
}
</div>
}

```



Calendar Events

May 4 2023
6:00 AM - 6:30 AM

Meet with Adatum Corporation about project schedule

January 4 2023
7:51 AM - 9:51 AM

Review Contract with Adatum Corporation

4. In addition to searching for calendar events using the `search/query` API, Microsoft Graph also provides an `events` API that can be used to search calendar events as well. Locate the `searchCalendarEvents()` function in `graph.service.ts`.
5. The `searchCalendarEvents()` function creates start and end date/time values that are used to define the time period to search. It then creates a URL that can be used to call the `events` endpoint of Microsoft Graph and includes the `query` parameter and start and end date/time variables. A GET request is then made and the results are returned to the caller.

TypeScript

```
async searchCalendarEvents(query:string) {
    if (!query) return [];
    const startDateTime = new Date();
    const endDateTime = new Date(startDateTime.getTime() + (7 * 24 * 60 * 60
* 1000));
    const url = `/me/events?
startdate=${startDateTime.toISOString()}&enddatetime=${endDateTime.toISOString()}&$filter=contains(subject,'${query}')&orderby=start/dateTime`;

    const response = await
Providers.globalProvider.graph.client.api(url).get();
    return response.value;
}
```

- Here's a breakdown of the URL that's created:
 - The `/me/events` portion of the URL is used to specify that the events of the signed in user should be retrieved.
 - The `startdate` and `enddatetime` parameters are used to define the time period to search. In this case, the search will return events that start within the next 7 days.
 - The `$filter` query parameter is used to filter the results by the `query` value (the company name selected from the datagrid in this case). The `contains()` function is used to look for the `query` value in the `subject` property of the calendar event.
 - The `$orderby` query parameter is used to order the results by the `start/dateTime` property.

Once the `url` is created, a GET request is made to the Microsoft Graph API using the value of `url` and the results are returned to the caller.

6. As with the previous components, the `calendar-events` component (`calendar-events.component.ts` file) calls `search()` and displays the results.

TypeScript

```
override async search(query: string) {  
    this.data = await this.graphService.searchCalendarEvents(query);  
}
```

ⓘ Note

You can make Microsoft Graph calls from a custom API or server-side application as well. View the [following tutorial](#) to see an example of calling a Microsoft Graph API from an Azure Function.

7. You've now seen example of using Microsoft Graph to retrieve files, chats, email messages, and calendar events. The same concepts can be applied to other Microsoft Graph APIs as well. For example, you could use the Microsoft Graph *users* API to search for users in your organization. You could also use the Microsoft Graph *groups* API to search for groups in your organization. You can view the full list of Microsoft Graph APIs [in the documentation](#).

Next Step

Congratulations

Congratulations

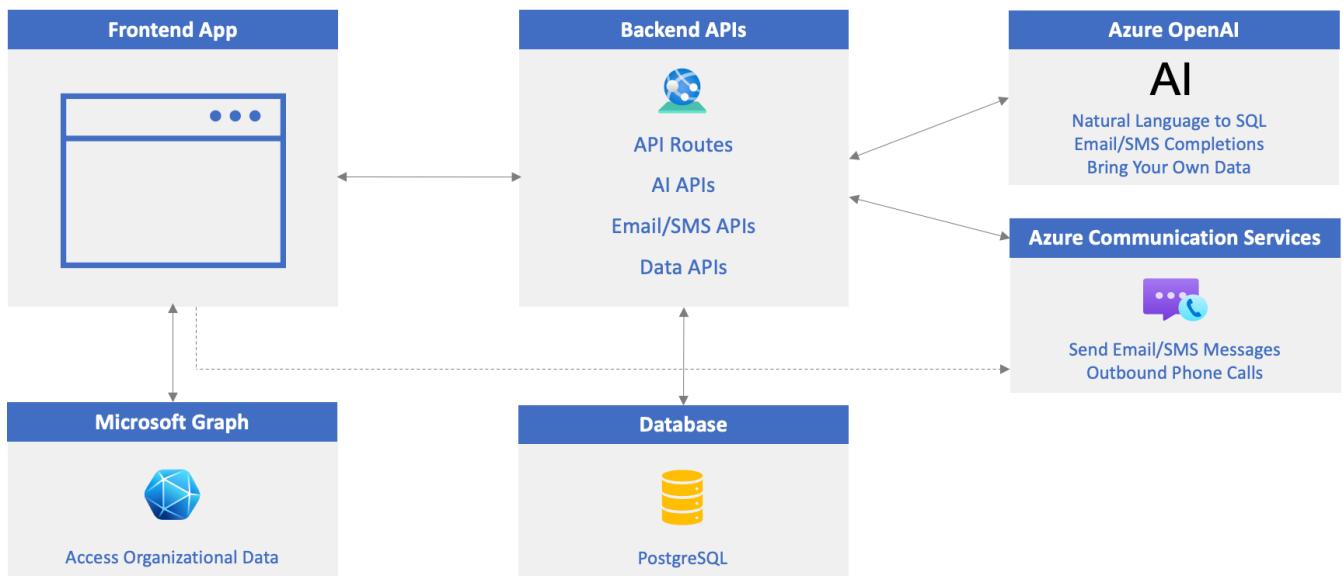
06/13/2025

You completed this tutorial

Congratulations! In this tutorial you learned how:

- Azure OpenAI can be used to enhance user productivity.
- Azure Communication Services can be used to integrate communication features.
- Microsoft Graph APIs and components can be used to retrieve and display organizational data.

By using these technologies, you can create effective solutions that increase user productivity by minimizing context shifts and providing necessary decision-making information.



Clean Up Azure Resources

Cleanup your Azure resources to avoid more charges to your account. Go to the Azure portal and delete the following resources:

- Azure AI Search resource
- Azure Storage resource
- Azure OpenAI resource (ensure that you delete your models first and then the Azure OpenAI resource)
- Azure Communication Services resource

Next Steps

Documentation

- [Azure OpenAI Documentation](#)
- [Azure OpenAI on your data](#)
- [Azure Communication Services Documentation](#)
- [Microsoft Graph Documentation](#)
- [Microsoft Graph Toolkit Documentation](#)
- [Microsoft Teams Developer Documentation](#)

Training Content

- [Apply prompt engineering with Azure OpenAI Service ↗](#)
- [Utilize an Azure OpenAI model to create an app](#)
- [Introduction to Azure Communication Services](#)
- [Microsoft Graph Fundamentals](#)
- [Video Course: Microsoft Graph Fundamentals for Beginners](#)
- [Explore Microsoft Graph scenarios for JavaScript development](#)
- [Explore Microsoft Graph scenarios for ASP.NET Core development](#)
- [Get started with Microsoft Graph Toolkit](#)
- [Build and deploy apps for Microsoft Teams using Teams Toolkit for Visual Studio Code](#)

What is Dev Proxy?

Article • 02/03/2025

Dev Proxy is an API simulator that helps you effortlessly test your app beyond the happy path.

You test your app to make sure it works as intended. But what if the APIs you use fail? Will your app lose your customer's data? How do you test for this? Simulating API failures is hard. You end up writing code that you won't be shipping or worse: not testing at all. That's why we built Dev Proxy, to simulate API errors so that you can easily test your app without changing your code.

With Dev Proxy you:

- [See how your app responds to API errors](#), without changing your app's code, so that you can **build more robust apps and don't lose customers' data**.
- [Verify how your app handles API rate limits](#), so that you can avoid getting throttled and **improve the user experience for your customers**.
- [See how your app handles slow APIs](#), so that you can implement the necessary affordances, and **make your app more user-friendly**.
- [Quickly stand-up mock APIs](#) without writing a line of code, so that you can **focus on building your app instead of writing code you won't be shipping**.
- Improve your app with contextual guidance on how you use APIs, to **make your app even better**.

Dev Proxy is a command-line tool that works on any platform. Because it intercepts network requests, it works with any type of app and tech stack. Dev Proxy is open source and free to use.

How does your app handle API errors?

[Get started](#)

Azure Architecture Center

Design solutions on Azure using established patterns and practices. Azure Architecture Center is a catalog of solution ideas, example workloads, reference architectures, technology decision guides, and architecture guides for Azure workloads.



ARCHITECTURE
Browse Azure architectures



CONCEPT
Build using application architecture...



REFERENCE
Make technology choices



CONCEPT
Learn cloud design patterns



WHAT'S NEW
See what's new and updated

Architect workloads on Azure

Explore best practices and patterns for building workloads on Microsoft Azure.



Design for the cloud

- Learn fundamental architecture styles
- Use best practices in cloud applications
- Avoid performance anti-patterns
- Use responsible engineering practices
- Use cloud design patterns



Choose the right technology

- Choose a compute service
- Choose a container service
- Choose a data store
- Choose AI services
- Choose a messaging service



Implement an Azure landing zone

- [Deploy a platform landing zone](#)
- [Bicep-based platform landing zone](#)
- [Terraform-based platform landing zone](#)
- [Automate through subscription vending](#)
- [Application landing zone solutions](#)

Specialized scenarios

- [Architecture for startups](#)
- [Architecture for SaaS and multitenant applications](#)
- [Build mission-critical workloads](#)
- [Azure for AWS professionals](#)
- [Azure for Google Cloud professionals](#)

Technology areas

Explore architectures and guides for different technologies.

Popular articles

- [AI Foundry chat baseline architecture](#)
- [Develop and optimize a RAG implementation](#)
- [AI agent orchestration patterns](#)
- [Cloud design patterns](#)
- [Hub-spoke network topology](#)
- [AKS production baseline](#)
- [Choose your Azure compute service](#)

AI & Machine Learning

- [AI architecture design](#)
- [AI Foundry chat baseline architecture](#)
- [Build a generative AI gateway](#)
- [AI agent orchestration patterns](#)
- [Develop and optimize a RAG implementation](#)
- [Use MLOps](#)
- [Use GenAIOps](#)

Analytics

- [Analytics architecture design](#)
- [Choose a data analytics technology](#)
- [Analytics end-to-end with Azure Synapse](#)
- [Enterprise BI with Microsoft Fabric](#)
- [Near real-time lakehouse data processing](#)
- [Stream processing with Azure Databricks](#)

Compute

- [Choose an Azure compute service](#)
- [Run a Linux VM on Azure](#)
- [Run a Windows VM on Azure](#)
- [Azure VM baseline architecture](#)
- [Understand multi-region compute balancing](#)
- [Build workloads on spot virtual machines](#)

Containers

- [Choose a container service](#)
- [Design for Azure Kubernetes Service](#)
- [Azure Kubernetes Service baseline cluster](#)

Databases

- [Databases architecture design](#)
- [Migrate an Oracle database to Azure](#)
- [DataOps for a modern data warehouse](#)

- Plan Day-2 operations on AKS
- Deploy microservices with Azure Container Apps
- Deploy microservices with Azure Container Apps and Dapr

- Design a medallion lakehouse
- Use change feed to replicate data

Hybrid + multicloud

- Hybrid architecture design
- Azure hybrid options
- Connect a cross-premises network to Azure
- Azure Local baseline architecture
- Hybrid Kubernetes clusters
- Hybrid file services

Identity

- Identity architecture design
- Identity in multitenant architectures
- Integrate on-premises AD with Microsoft Entra ID
- Extend AD DS to Azure
- Create an AD DS forest in Azure

Networking

- Choose a hybrid network architecture
- Hub-spoke topology
- ExpressRoute with VPN failover
- Implement a secure hybrid network
- Highly available network virtual appliances

Security

- Security architecture design
- Azure security in AWS
- Zero-trust network with Azure Firewall and Application Gateway
- Securely managed web applications

Web apps

- Enterprise web app patterns
- Basic web application
- Baseline zone-redundant web application
- Protect APIs with Application Gateway and API Management
- Serverless web application

Cloud adoption and workload design

Build a strong cloud adoption strategy and a consistent approach to workload design.



[Cloud Adoption Framework for Azure](#)



[Azure Well-Architected Framework pillars](#)



[Well-architected workloads](#)



[Well-architected service guides](#)

Azure developer documentation

Find the languages and tools you need to develop on Azure.



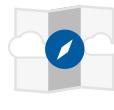
CONCEPT
[What's new in Azure developer documentation](#)



GET STARTED
[Introduction to Azure for developers](#)



OVERVIEW
[Key Azure services for developers](#)



OVERVIEW
[Host applications on Azure](#)



OVERVIEW
[Connect apps to Azure services](#)



OVERVIEW
[Develop apps that use Azure AI services](#)



CONCEPT
[Create resources in Azure](#)



ARCHITECTURE
[Key concepts for building Azure apps](#)

GitHub Copilot for Azure Preview

[What is GitHub Copilot for Azure?](#)

[Get started](#)

[Quickstart](#)

[Get information about Azure and your apps](#)

[Deploy your apps](#)

[Troubleshoot your apps](#)



Azure MCP Server

[What is the Azure MCP Server?](#)

[Get started with GitHub Copilot with Visual Studio Code](#)

[Get started with .NET](#)

[Get started with Python](#)

[Azure MCP Server tools](#)



Python

[Deploy serverless Python apps to Azure Functions](#)

[Deploy Python apps to Azure App Service](#)

[Manage storage blobs with the Azure SDK for Python](#)

[Use Python to query Azure SQL Database](#)

[Create an Azure Data Factory using Python](#)

[See more in the Python developer center](#)



JavaScript

[Develop Azure AI apps with JavaScript](#)

[Generative AI for JavaScript developers](#)

[Deploy a serverless application](#)

[Azure Container Apps for Node.js](#)

[Deploy to Azure App Service](#)

[See more in the JavaScript developer center](#)



Java

[Install the JDK for Azure and Azure Stack](#)

[Deploy an app to Azure Spring Apps by using the Azure portal](#)

[Create a Java app in Azure App Service](#)

[Use Spring Boot Starter for Microsoft Entra ID](#)

[Migrate Java Applications to Azure](#)

[See more in the Java developer center](#)



.NET

[Introduction to Azure and .NET](#)

[Configure your .NET development environment for Azure](#)

[Deploy an ASP.NET web app](#)

[Build a serverless function](#)

[Azure SDK for .NET](#)

[See more in the .NET developer center](#)

Azure AI for developers

[Overview of AI apps](#)
[Chat with your data using Python](#)
[Chat with your data using JavaScript](#)
[Chat with your data using Java](#)
[Chat with your data using .NET](#)
[Azure AI services documentation](#)

Go

[Install the Azure SDK for Go](#)
[Authenticate your app](#)
[Develop Azure AI apps with Go](#)
[Use Blob storage](#)
[Azure SDK for Go code samples ↗](#)
[See more in the Go developer center](#)

C++

[What is the Azure SDK for C++?](#)
[Install the Azure SDK for C++](#)
[Authentication in the Azure SDK for C++](#)
[SDK fundamentals](#)
[Common types in the Azure SDK for C++](#)
[See more in the C++ developer center](#)

Rust

[Rust on Azure](#)
[Set up development environment](#)
[Azure SDK for Rust](#)
[Install the Azure SDK](#)
[Authenticate your app](#)
[See more in the Rust developer center](#)

Azure PowerShell

[What is the new Az module?](#)
[Migrate from AzureRM to Az](#)
[Install](#)
[Sign in](#)
[Persist credential contexts](#)
[See more in the Azure PowerShell developer center](#)

Azure CLI


[Get started](#)
[Install](#)
[Sign-in](#)
[Run a query on the output of a CLI command](#)
[Create a virtual machine](#)
[See more in the Azure CLI developer center](#)



Azure Developer CLI (azd)

[What is the Azure Developer CLI? \(azd\)](#)

[Install azd](#)

[Deploy an azd template](#)

[Make your project compatible with azd](#)

[Supported languages and environments](#)

TypeSpec

[Overview](#)

[Get started](#)

[Language reference](#)

[Playground](#)

[Resources](#)

Dev Proxy

[What is Dev Proxy?](#)

[Get started](#)

[Test a JavaScript client-side web application](#)

[Test a JavaScript client-side web application that calls Microsoft Graph](#)

[How to implement rate limiting in Azure API Management](#)

Developer Tools

Use your favorite development tools when working with Azure

[Visual Studio](#) ↗

Full-featured IDE to code, debug, test, and deploy to any platform.

[Visual Studio Code](#) ↗

Free. Built on open source. Runs everywhere.

[Visual Studio Code Azure Extensions](#) ↗

Use Azure directly from Visual Studio Code through extensions.

[DevOps and Automation](#)

Automate deployment with Infrastructure as Code (IaC).

[Playwright](#)

Reliable end-to-end testing for modern web apps

[Azure Toolkit for IntelliJ](#)

Use Azure Toolkit for IntelliJ to create, develop, test, and deploy Azure applications from within the IntelliJ IDE.

[Azure Toolkit for Eclipse](#)

Use Azure Toolkit for Eclipse to create, develop, test, and deploy Azure applications from within the Eclipse IDE.

[Maven](#)

Use Maven to automate the way you build and manage Java projects. It works with other languages too.

Azure Government documentation

Azure Government is a cloud services platform built upon the foundational principles of security, privacy, and compliance. Customers eligible for Azure Government benefit from a physically isolated instance of Azure.

About Azure Government

OVERVIEW

[What is Azure Government?](#)

[Compare Azure Government and global Azure](#)

Get started

QUICKSTART

[Connect using portal](#)

[Connect using Azure CLI](#)

[Connect using PowerShell](#)

[Connect using Visual Studio](#)

Compliance

TRAINING

[Azure Government compliance](#)

[Cloud services by audit scope](#)

[Trusted Internet Connections](#)

[Accelerate your path to ATO](#)

Deploy and manage

TUTORIAL

[Deploy an Azure App Services app](#)

[Create virtual machines](#)

[Deploy with Azure Pipelines](#)

[Use DISA CAP](#)

[Monitor with Azure Monitor logs](#)

[Azure Government Marketplace](#)

How-to guides

HOW-TO GUIDE

[Guidance for developers](#)

[Impact Level 5 isolation guidance](#)

[Azure secure isolation guidance](#)

[Secure Azure computing architecture](#)

[Develop with Storage API](#)

[Develop with Azure AI services](#)

[Integrate Microsoft Entra authentication](#)

[Develop with SQL Server Management Studio](#)

[Configure Microsoft cloud services for the DoD Zero Trust Strategy](#)

Concepts

CONCEPT

[Considerations for naming Azure resources](#)

[Security](#)

[Identity](#)

[Department of Defense](#)

[Export controls](#)

[Power and utilities](#)

[Public safety and justice](#)

Resources

REFERENCE

[Marketplace image gallery](#)

[Marketplace VM extensions](#)

[Marketplace for partners](#)

[CSP application process](#)

[Authorized reseller list](#)

Developer resources for building and extending Dynamics 365 apps

Article • 08/30/2024

Dynamics 365 is a suite of intelligent business apps that lets you engage with customers and build relationships, optimize operations, connect and grow your businesses, and empower employees by attracting and hiring the best talent.

Developers and ISVs can use the underlying platform to build or extend apps, depending on the app's type. For information about different types of Dynamics 365 business apps, see [What's new](#).

Start here to find information about building and extending Dynamics 365 apps.

[] Expand table

Dynamics 365 app	Resources
Business Central	<ul style="list-style-type: none">- Development and administration- Get Started with Building Apps- Development in AL
Dynamics 365 Finance Dynamics 365 Supply Chain Management	<ul style="list-style-type: none">- Develop and customize- User interface development- Mobile platform- Develop and release solutions using Lifecycle Services- Financials development
Dynamics 365 Sales Dynamics 365 Customer Service Dynamics 365 Field Service Dynamics 365 Marketing Related model-driven apps	<ul style="list-style-type: none">- Model-driven apps Developer Guide- Publish your app on AppSource
Dynamics 365 Commerce	<ul style="list-style-type: none">- Commerce development and extensibility
Dynamics 365 Human Resources	<ul style="list-style-type: none">- Embed Power Apps- Human resources and Dataverse tables

Related information

- [Microsoft AppSource](#)
- [Microsoft Power Platform](#)

- Common Data Model
- Training resources
- Roles in Dynamics 365

Microsoft 365 developer documentation

Build enterprise-grade agents and apps with the integrated Microsoft 365 platform and unlock AI-powered productivity.



WHAT'S NEW

[What's new for Microsoft 365
Copilot extensibility](#)



GET STARTED

[Build agents for Copilot](#)



OVERVIEW

[Bring Copilot to your agents and apps](#)



DOWNLOAD

[Get Microsoft 365 Agents Toolkit](#)

Explore the integrated Microsoft 365 platform



[Microsoft 365 Copilot extensibility](#)

Build agents and connectors that extend Copilot with your business skills and knowledge.



[Microsoft Teams platform](#)

Build agents and apps that work in Teams, Copilot, and Outlook.



[Office Add-ins](#)

Extend Office applications to interact with content in Office documents and Outlook mail and calendar items.



[Microsoft Graph](#)



[SharePoint Framework](#)



[Power Apps](#)

Build intelligent apps, derive insights and analytics, and extend Microsoft 365 experiences.

Customize and extend SharePoint experiences across Microsoft 365, in Microsoft Viva, Teams, Outlook, and...

Build low-code apps that modernize processes and solve tough business challenges in your organization.

Tools and libraries



[Microsoft 365 Agents Toolkit](#)



[Microsoft 365 Agents Toolkit CLI](#)



[Microsoft 365 Agents SDK](#)



[Microsoft 365 Copilot APIs](#)



[Copilot Studio in Microsoft 365 Copilot](#)



[Copilot Studio](#)



[Microsoft 365 Developer Program](#)



[Microsoft Teams SDK \(AI Library\)](#)



[Microsoft Teams CLI](#)



[Adaptive Cards](#)



[Microsoft Graph Explorer](#)



[Microsoft 365 app manifest reference](#)

Related content

[Microsoft 365 Adoption](#)

Find resources for business decision makers

[Microsoft 365 for Administrators](#)

Find resources for IT admins and security professionals

[Microsoft 365 Help & Learning](#)

[Microsoft 365 Dev Center](#)

Support for Microsoft 365 users

Microsoft 365 platform highlights and key resources

Find resources

Publishing your app

[Microsoft 365 Store publishing checklist](#)

[Microsoft Commercial Marketplace certification policies](#)

[Microsoft 365 App Compliance Program](#)

Community

[Microsoft 365 developer blog ↗](#)

[Microsoft 365 developer videos ↗](#)

[Microsoft 365 community videos ↗](#)

[Microsoft 365 platform community ↗](#)

Learning paths and training

[Microsoft 365 Copilot training](#)

[Extend Microsoft 365 Fundamentals](#)

[Microsoft 365 Certifications](#)

Feedback and help

[Microsoft 365 Roadmap ↗](#)

[Microsoft 365 feedback portal ↗](#)

[Microsoft 365 Q&A community support](#)

Microsoft industry documentation and resources

Microsoft offers a range of solutions, guidance, and resources tailored to your industry. This documentation includes information about the Microsoft for Healthcare, Financial Services, Retail, Manufacturing, Nonprofit, Sovereign Cloud, and Sustainability.

Microsoft Industry Solutions

Microsoft Industry Solutions are built for your industry, combining Microsoft Azure, Microsoft 365, Microsoft Dynamics 365, Microsoft Fabric, Microsoft Power Platform, and agentic AI.



Financial Services

Find information about the Microsoft for Financial Services and other solutions for the Financial Services industry....



Healthcare

Find information about Microsoft for Healthcare and other solutions for the Healthcare industry. Microsof...



Manufacturing

Microsoft for Manufacturing is designed for delivering capabilities that support the core processes and...



Nonprofit

Nonprofit industry solutions are designed to accelerate the creation, configuration, and customization of solutions on...



Retail

Find information about Microsoft for Retail and other solutions for the retail industry. Help connect your customers,...



Digital sovereignty

Microsoft Sovereign Cloud offers a comprehensive spectrum of digital sovereignty capabilities, across integrated...



Sustainability



Reference architecture center



Well-Architected for Industry

Help your company take control of your environmental initiatives with this set of comprehensive, integrated,...

Explore patterns, best practices, and detailed diagrams for learning more about successfully deploying...

Improve the quality of your Microsoft Cloud for industry workloads with comprehensive checklists, design principles,...

Product-specific architectures and solutions for your industry

Find example architectures, solution ideas, and related guidance for your industry created for Microsoft Azure, Microsoft Dynamics 365, Microsoft Power Platform, and Microsoft 365.

Mobility

-  Microsoft solutions for mobility and software-defined vehicles

Energy and resources

-  Azure solutions for the energy and environment industries
-  Key Compliance and Security Considerations for the Energy Industry
-  Azure Data Manager for Energy

Government

-  Azure solutions for Government
-  Microsoft Teams for Government
-  Training for government

Media and Telecommunications

-  Azure solutions for the media and entertainment industry
-  Dynamics 365 media and entertainment accelerator
-  Dynamics 365 telecommunications accelerator

Agriculture

-  Azure Data Manager for Agriculture

Apps, solutions, and connectors

Find apps, solutions, and connectors to use Microsoft products in your industry.

[Microsoft Azure Marketplace](#)

Discover, try, and deploy the cloud software you want.

[Microsoft AppSource](#)

Find apps for your industry from Microsoft and partners - compatible with the Microsoft Industry Clouds.

[Microsoft Fabric](#)

Learn about Microsoft Fabric, evaluate its features, the experiences it offers, analyze your data and run reports.

[Industry Solutions in Microsoft Fabric](#)

Integrate different data sources for analytics and reporting.

[Power Platform connectors](#)

Learn about Copilot Studio, Power Platform, and Azure Logic Apps connectors.

[Copilots, agents, and generative AI in Industry Solutions](#)

Learn about copilots, agents, and generative AI in Industry Solutions.

Trust Center, Partner program, architecture resources, and customer stories

Find resources and ideas on how to use Microsoft products in your industry.

[Microsoft Trust Center](#)

Learn more about the commitments we've made to ensure your data is safeguarded, our practices ar...

[Microsoft Cloud Partner Program](#)

Explore the Microsoft AI Cloud Partner Program for the tools to create and market innovative Microsoft Cloud a...

[Azure Architecture Center](#)

Design solutions on Azure using established patterns and practices.

[Microsoft 365 solution and architecture center](#)

Understand, plan, and implement integrated Microsoft 365 solutions for enterprise resource planning...

[Customer stories](#)

Learn about what worked for other customers - filter customer stories by industry.

Microsoft Power Platform developer documentation

Learn how to extend Microsoft Power Platform using code first components and integrating with back-end services.

Get started



[OVERVIEW](#)

[Power Platform for developers](#)



[HOW-TO GUIDE](#)

[Create a developer environment](#)



[TRAINING](#)

[Introduction to developing with Power Platform](#)

Fusion development



[CONCEPT](#)

[Fusion development](#)



[REFERENCE](#)

[Fusion development ebook](#)



[TRAINING](#)

[Transform business apps with fusion development](#)

ALM for developers



[REFERENCE](#)

[Power Platform ALM basics](#)

[Build Tools for Azure DevOps](#)

[GitHub Actions](#)

[Power Platform CLI](#)

[Create packages using Package Deployer](#)

[Work with solutions using SDK](#)

For ISVs



HOW-TO GUIDE

[Publish your app on AppSource](#)

Tools for Developers



DOWNLOAD

[Power Platform CLI](#)

[Power Platform Tools for Visual Studio](#)

[ALM developer tools](#)

[Dataverse developer tools](#)

[Visual Studio connected service](#)

Dataverse



REFERENCE

[Dataverse developer docs](#)

[Plug-ins](#)

[Webhooks](#)

[Custom APIs](#)

[Virtual tables](#)

Power Apps

REFERENCE

[Canvas apps developer docs](#)

[Model-driven apps developer docs](#)

[Power Apps component framework](#)

[Custom connectors](#)

[JavaScript and the Client API](#)

[Code apps \(preview\)](#)

QUICKSTART

[Create a Canvas App front end for your ASP.NET web API \(Preview\)](#)

Power Automate

REFERENCE

[Custom connectors](#)

[Use workflow definition functions](#)

[Use APIs with desktop flows](#)

Power BI

REFERENCE

[Custom visuals](#)

[Use Power BI REST APIs](#)

[Analysis Services developer documentation](#)

Copilot Studio



[Bot Framework Composer](#)

Power Pages



[Custom page layouts](#)

[Work with Liquid templates](#)

[Power Pages Web API](#)

[Use Visual Studio Code and Power Platform CLI](#)

[Use code components](#)