PL-900: Microsoft Power Platform Fundamentals

Power Apps

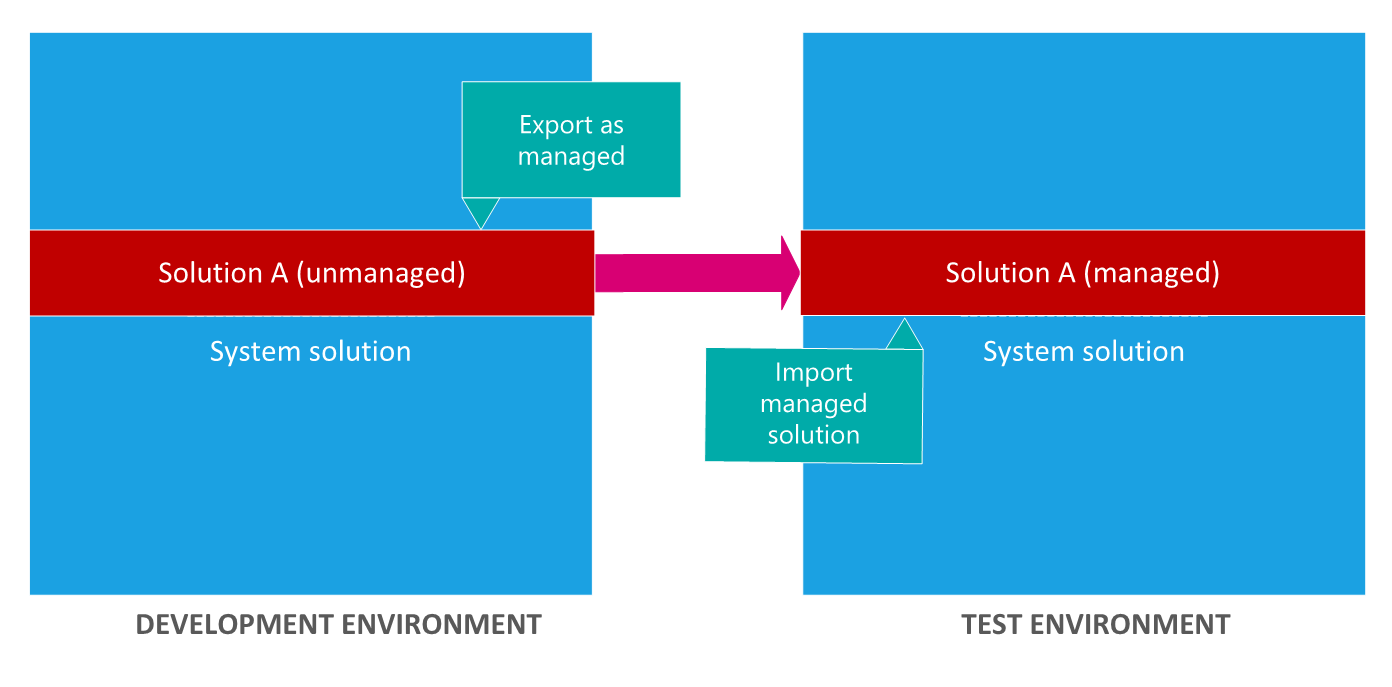
**Environment** – can be created from “Setting” -> “Admin Center”, then go to “Environments” on left menu. Click on + sign to create a new environment. Each environment consume 1GB of space with or without Dataverse.

In environment, you can select “Region”. Select “Type” such as Production, Developer, Trial, Sandbox. Trial environment valid for 30 days and automatically remove after that.

**Dataverse** – can store business rules, not just storing the data. Business rules can be things like “trigger” if a record is stored in table or etc.

**Tables** – also called entities or objects. You have default tables and custom tables which can be also imported from unmanaged solutions (include customization created in dev env).

Read more at <https://learn.microsoft.com/en-us/power-platform/alm/solution-concepts-alm>



**Unmanaged solutions** are developed and **managed solutions** are deployed. Can’t edit managed solutions directly. If an unmanaged solution is deleted, all customizations will go under default solution and will not be removed. But if the managed solution is deleted, all customizations will be gone.

**Table types** – Standard, Activity (include actions such as Email), Virtual and Elastic

**Row ownership** – two types of row ownerships 1) own by user or team 2) own by organization.

**Table Relationship** – support three types: 1-Many, Many-1, Many-Many.

**Business Rule** – this was applied to Table. Business Rule allow us to add business logic. For example, if the user input value is something, then we can do something to some fields in the table. Some form of validation. Model driven app provides more options for business rule compared to Canvas app.

**Realtime workflow** – workflows are added to the solution. Go to solution and select the solution you want to add workflows. Workflow is located under Automation -> Process. Workflows are attached to tables (entity). Example, before or after a row status change in table, and such situation, we can add “Steps” to perform task. Steps can group into “Stages”. Tips: make sure to uncheck “run in background” if we want the workflow to run real-time.

**Actions** – Like workflow. The difference is action not required to attached to table. Action run based on something happened, such as workflow executed. Action provide messages which we can pass the variables and send to third parties, then process it and return value also can be captured in that action message.

**Dataflow** – allow to create a process. Many connectors to connect the data (such as Excel workbook or online data sources, on-prem resources need data gateway to connect). Useful to create ETL process using this. Example, extracting data from a data sources, transform it and load it back into warehouses.

**More on Dataverse table**

A screenshot of a computer

Description automatically generated

**Views** – each table has many views, and we can customize them, such as adding columns.

**Forms** – to input data into table. There are four different types of forms.

1. Main – you must have it for every table
2. Quick View – provides additional data from the table, e.g., detail view, display data
3. Quick Create – basic form to create new row, input data
4. Card – compact form usable for mobile devices serves same as Quick View

Power Automate

**Instant** – runs on demand on button press from power app or another place. e.g., notification.

**Automated** – runs based on system event, e.g., storing email attachment to One Drive.

**Scheduled** – runs on schedule., e.g., daily reminder email and this reoccurs.

Power Automate use triggers. Power App can be a source to make use of the Power Automate triggers.

## Introduction to developing with Microsoft Power Platform

Five main components: Power Apps, Power Automate, Power BI, Power Pages, and Power Virtual Agents

Graphical user interface, application

Description automatically generated

1. Power Apps Learning Path:

<https://learn.microsoft.com/en-us/training/paths/create-powerapps/>

<https://learn.microsoft.com/en-us/training/paths/create-app-models-business-processes/>

1. Power Automate Learning Path:

<https://learn.microsoft.com/en-us/training/paths/automate-process-using-flow/>

<https://learn.microsoft.com/en-us/training/paths/pad-get-started/>

1. Power BI Learning Path

<https://learn.microsoft.com/en-us/training/paths/create-use-analytics-reports-power-bi/>

1. Power Pages Learning Path

<https://learn.microsoft.com/en-us/training/paths/power-pages-get-started/>

1. Power Virtual Agents Learning Path

<https://learn.microsoft.com/en-us/training/paths/work-power-virtual-agents/>

Connector - is a proxy or a wrapper around an API that allows the underlying service to talk to Microsoft Power Automate, Microsoft Power Apps, and Azure Logic Apps. It provides a way for users to connect their accounts and use a set of prebuilt actions and triggers to build their apps and workflows.

Dataverse – cloud-scale data store allows RBAC on tables and at rows level.

Common Data Model - an open-sourced standard definition of entities that represent commonly used concepts and activities. Tips: Use the Common Data Model Library to work with data stored in Azure Data Lake. Refer to documentation [here](https://learn.microsoft.com/en-us/common-data-model/schema/core/applicationCommon/overview/).

Graphical user interface, application

Description automatically generated

**Common Data Model** is influenced by data schemas that are present in Dynamics 365, covering a range of business areas.

The Common Data Model is maintained in [GitHub](https://github.com/microsoft/CDM/). Schemas are maintained as JSON files. An entire entity reference can be found at [Overview of applicationCommon](https://learn.microsoft.com/en-us/common-data-model/schema/core/applicationCommon/overview/). Microsoft also provides a [Visual Entity Navigator](https://microsoft.github.io/CDM/) that allows users to visually navigate Common Data Model entities.

Extensibility - any code-based customization activities are referred to as "extending" an application. To extend the user experience of a Power Apps app, you can use Power Apps Component Framework (PCF) to create code components that are used across all Power Apps types. Model-driven Power Apps also exposes a JavaScript client API that allows you to interact with application pages to implement business logic on the application forms.

While canvas apps don't offer scripting capabilities, their logic is built using [Microsoft Power Fx](https://learn.microsoft.com/en-us/power-platform/power-fx/overview/). Power Fx is the low-code language that is used across Microsoft Power Platform. It's a general-purpose, strong-typed, declarative, and functional programming language.

Extending Dataverse - need for some level of automation or behaviour not supported by the existing features. These types of extensibility points run as server-side code through a construct called a plug-in. When requirements for synchronous operations exist within an application, plug-ins are still required. Synchronous processing by plug-ins allows any Dataverse modifications to be treated as a single unit of work in a transaction.

Dataverse Solutions - Solutions are the mechanism for implementing application lifecycle management (ALM) in the Power Platform. For detailed information about the solution concepts, see [Solutions in Power Apps](https://learn.microsoft.com/en-us/power-apps/maker/data-platform/solutions-overview/) in the Dataverse documentation. Power Platform build tools can be used with Azure Dev Ops or GitHub to automate Power Platform ALM operations. For more information how solutions are used for application lifecycle management, see [Overview of ALM with Microsoft Power Platform](https://learn.microsoft.com/en-us/power-platform/alm/overview-alm/) in Microsoft Power Platform ALM guide.

### Extending Power Platform with Azure

Azure cloud services provide a wealth of functionality and enable Microsoft Power Platform developers to harness its capabilities through various extensibility points.

Azure Functions - is a supported extensibility endpoint for Power Platform. Use it to implement business logic outside of apps. Build APIs that are exposed to Microsoft Power Platform as a **custom connector**. As a custom connector the Azure Function business logic can easily be invoked from apps and flows.

API Management - manage APIs across clouds and on-premises. In addition, API Management can export API definitions directly to Microsoft Power Platform. When exported, the API is configured as a **custom connector** available in Power Apps and Power Automate.

Service Bus - reliable [messaging-as-a-service](https://azure.microsoft.com/services/service-bus/) (MaaS) framework that enables real-time, asynchronous messaging across systems. Developers can configure Microsoft Dataverse to publish events to Azure Service Bus queues and topics. Events can be published automatically on Dataverse data modifications or on demand from developer's custom logic. Service bus can store the message until the consuming party is ready to receive the messages allowing you to architect solutions that are less dependent.

Event Grid - fully managed [single service for managing routing for all events](https://azure.microsoft.com/services/event-grid/) from any source for any destination. It simplifies the development of event-based applications and the creation of serverless workflows. Event Grid can be used to route events between Microsoft Power Platform and other Azure services like Azure Functions.

Logic Apps - a [cloud service](https://azure.microsoft.com/services/logic-apps/) that helps you schedule, automate, and orchestrate tasks, business processes, and workflows when you need to integrate apps, data, systems, and services across enterprises or organizations.

AI Services - a family of AI and cognitive APIs to help build intelligent apps. Azure AI Services can be categorized into five main pillars: vision, speech, language, web search, and decision.

Azure Data Lake and Azure Synapse analytics - Microsoft Power Platform data seamlessly supports [Azure Data Lake storage](https://azure.microsoft.com/services/storage/data-lake-storage/) with its Microsoft Dataverse and Common Data Model framework. Any data or metadata changes in Dataverse are automatically pushed to the Azure Synapse metastore and Azure Data Lake, depending on the configuration, without any additional action. This is a push, rather than pull, operation. Changes are pushed to the destination without you needing to set up refresh intervals.

Azure SQL Database - Dataverse itself is built on top of Azure SQL Database engine. Power Apps and Power Automate can work with data in Azure SQL via the Azure SQL connector.