# Program Connections

Programming connections is an essential aspect of deploying Fabric solutions. For example, you will be required to write code to create and manage connections when deploying Fabric solutions which include workspace items such as OneLake shortcuts, data pipelines and semantic models.

Connections in Fabric are similar to workspaces in the sense that they are both platform items as opposed to workspace items. They are scoped to the current Entra Id tenant. You configure access to connections by adding connection role assignments.

## Connection Fundamentals

As a Fabric developer, it’s important for you to distinguish between inbound security versus outbound security. **Inbound security** is involved when a custom application running outside the Fabric environment executes API calls on Fabric REST API endpoints. As you learned in earlier chapters, a custom application must first authenticate with the Entra Id Service and acquire access tokens before it can securely call the Fabric REST APIs. The custom application must then transmit an access token in each and every API call executed on a Fabric REST API endpoint.

**Outbound security** is different than inbound security. That’s because outbound security involves a scenario where you’ve created some type of workspace item running inside the Fabric environment which must connect to an external datasource. For example, you can create a OneLake shortcut which connects to an ADLS Gen2 storage container. In another example, you can create a semantic model which connects to an Azure SQL database.

A close-up of a screen

Description automatically generated

There are three types of workspace items that use connections.

* OneLake shortcuts
* Semantic models
* Data pipelines

Keep in mind that when you create a connection, you’re not really establishing a connection across the network at that point in time. Instead, you are really just creating a persistent Fabric object with metadata for creating connections at some point in the future. The metadata that Fabric persists for a connection includes the datasource type and location. The persisted metadata for a connection also includes whatever security credentials are required to authenticate and gain access to the external datasource.

So when is the persisted metadata for a connection actually used establish a connection? It is when a workspace item such as a OneLake shortcut is accessed. Think about a scenario when a user opens a lakehouse in the browser-based Fabric user experience and examines a shortcut which is bound to a Fabric connection for an ADLS Gen2 storage container. When the user clicks on the shortcut to examine the files in the ADLS Gen2 storage container, that will trigger the Fabric environment to read the persisted connection metadata to establish a true connection to the datasource across the network.

### Connectors

Every Fabric connection is based on a specific connector. A connector is a component containing the logic to connect to a specific type of datasource. For example, Fabric uses the **AzureDataLakeStorage** connector to create connections and access content in an ADLS Gen2 storage container. Fabric uses the **SQL** connector to create connections and access content in an Azure SQL Database.

A close-up of several red boxes

Description automatically generated

Fabric currently supports more than 200 different connectors. The list of supported connectors and their capability is growing constantly. Fabric provides the **List Supported Connection Types** API which makes it possible for a developer to discover which connectors are available. The metadata that the **List Supported Connection Types** API returns for each connector also allows the developer to discover what parameters need to be passed when creating a connection based on that connector as well as which types of credentials that the connector supports.

### Connection Connectivity Types

Fabric connections are split out into separated categories based on their connectivity type. The **connectivity type** is a connection property that indicates the architectural details of whether connection relies on a gateway as well as which type of gateway. There are two cloud-based connectivity types which do not require a gateway. There are three more connectivity types that each require a specific type of gateway.

1. **ShareableCloud**  - Shareable cloud connection
2. **PersonalCloud** - Personal cloud connection
3. **VirtualNetworkGateway** - Virtual gateway connection
4. **OnPremisesGateway**  - On-prem gateway connection
5. **OnPremisesGatewayPersonal**  - Personal on-prem gateway connection

**Shareable cloud connections** are what Microsoft recommends when connecting to datasources that do not require a gateway. As its name implies, a shareable cloud connectioncan be shared with other users and service principals using role assignments. The creator of a shareable cloud connectionis automatically assigned the **Owner** role. You can configure other users and service principals with access by adding connection role assignment based on the roles of **User**, **User with Reshare** or **Owner**.

**Personal cloud connections** represent the original type of connection used in Power BI since its inception. The main problem with personal cloud connections is that they cannot be shared. Each personal cloud connection is exclusively owned and used by a single user or service principal. Personal cloud connections should be considered as a legacy type of connection that should only be used in scenarios which require backwards compatibility.

Programming gateways and gateway connections are covered in a later chapter. This chapter is exclusively focused on building your understanding of shareable cloud connections and creating them using the Fabric REST APIs.

### Credential Types

When creating connections using the Fabric REST APIs, the creation request must include credentials supported by the target datasource. Fabric supports creating connections using the following credentials types.

* Anonymous
* Basic
* OAuth2
* Key
* SharedAccessSignature
* ServicePrincipal
* Windows
* WindowsWithoutImpersonation
* WorkspaceIdentity

Encryption Types can have a value of either NotEncrypted or Encrypted

Skip Connection Test

Single signon type

### Create Connections by Hand

Before you attempt to automate the creation of connections using code, it can be helpful to create a few connections by hand. This exercise will give you a sense of what decisions you need to make whenever you need to create a new connection.

Remember that connections are platform items that exist within the scope of the current Entra Id tenant. That means that Fabric connection can be shared across workspaces. Of course, just because you can doesn’t mean that you should.

You can see all the connections you have access to using the Manage Connections and Gateways page in the Fabric Service.

A screenshot of a computer

Description automatically generated

Dddddd

A screenshot of a computer

Description automatically generated

Xxxx

A screenshot of a computer

Description automatically generated

Xxxx

A screen shot of a computer

Description automatically generated

xxx

A screenshot of a computer

Description automatically generated

xxxx

### Supported Connection Types

### Creation Methods and Parameters

Creation method defines a name and set of parameters.

There are many different connection types. They are supported by underlying connects.

## Program CRUD Operations for Connection

### List Connections

### Create Connection

Ccccc

### Get Connection

### Update Connection

### Delete Connection

## Program Connection Role Assignments

### Add Connection Role Assignment

### List Connection Role Assignments

### Delete Connection Role Assignment

### Create an Anonymous Web Connection

Content to come

### Create a Azure SQL Connections using Basic Credentials

Content to come

### Create an Azure Storage Connection using Service Principal Credentials

Content to come

## Discover Supported Connection Types