📦 Step 1: Install the NuGet Package

Start by installing the **AzureServiceBusFlow** package from NuGet:

```
dotnet add package AzureServiceBusFlow
```

You can also find it on NuGet.org.

This package contains all the necessary components to configure Producers / Consumers, register Queues / Topics and publish Messagens / Events to AzureServiceBus.

⚙️ Step 2: Configure the `appsettings.json`

After installing the package, it's time to configure your `appsettings.json` file.
You need to define the basic configuration for Azure Service Bus to work.

Here's a configuration example:

```
"AzureServiceBusConfigurationSettings": {
    "ConnectionString": "",
    "ServiceBusReceiveMode": "ReceiveAndDelete",
    "MaxAutoLockRenewalDurationInSeconds": "1800",
    "MaxConcurrentCalls": "10",
    "MaxRetryAttempts": "3"
}
```

- **ConnectionString**: The full connection string to your Azure Service Bus instance, typically in the format: `Endpoint=sb://<your-servicebus-namespace-name>.servicebus.windows.net/;SharedAccessKeyName=<your-access-key-name>;SharedAccessKey=<your-access-key-value>`.
- **ServiceBusReceiveMode**: Defines how messages are handled after being received.
  - **"PeekLock"**: The message is locked for processing and remains in the queue until it is explicitly completed or abandoned. If the receiver fails to complete the message within the lock duration, the message becomes available again for other receivers.
  - **"ReceiveAndDelete"**: The message is automatically removed from the queue as soon as it is received.
- **MaxAutoLockRenewalDurationInSeconds**: The maximum duration, in seconds, that the message lock will be automatically renewed while the message is being processed.
- **MaxConcurrentCalls**: Specifies the maximum number of messages that can be processed concurrently.

- **MaxRetryAttempts**: Defines the maximum number of retry attempts the message handler will perform to reprocess a message if an exception occurs.

🧩 Step 3: Register AzureServiceBus in `Program.cs`

Now that your `appsettings.json` is configured, it's time to enable AzureServiceBus in your application.

Inside your `Program.cs`, register the AzureServiceBus in DI container with the AddAzureServiceBus extension method and set the configure method.

```
builder.Services.AddAzureServiceBus(cfg => cfg
    .ConfigureAzureServiceBus(azureServiceBusConfig));
```

This method need a **`AzureServiceBusConfiguration`** instance to configure all the necessary properties to work with the Azure Service Bus. This configuration class is provided by **AzureServiceBusFlow** and its parameters are the same as the JSON object configured in ⚙️ **Step 2**.

At this time, no queues, topics, producers or consumers are configured and registered. This will be done in the next pages.

> 💡 I recommend you create a class called Settings and map the appsettings.json on this class, you can merge your personal appsettings config and also add the AzureServiceBusFlow properties that are required. For example:

```
{
  "Settings": {
    "AzureServiceBusConfigurationSettings": {
        "ConnectionString": "",
        "ServiceBusReceiveMode": "ReceiveAndDelete",
        "MaxAutoLockRenewalDurationInSeconds": "1800",
        "MaxConcurrentCalls": "10",
        "MaxRetryAttempts": "3"
    }
  }
}
```

```
public class Settings
{
    public AzureServiceBusConfiguration AzureServiceBusConfigurationSettings { get; }
}
```

```
var applicationSettings = builder.Configuration.GetSection("Settings").Get<Settings>();
```

🔒 Next Steps: Create Messages, Producers, Consumers, Queues and Topics.

In the next step, we will explain how to create Messages and Producers that will publish to a specific queue or topic in Azure Service Bus.

Namespace AzureServiceBusFlow.Abstractions

Interfaces

ICommandProducer<TCommand>

IEventProducer<TEvent>

IMessageHandler<T>

IServiceBusMessage

IServiceBusProducer<TMessage>