Getting Started with the C# SDK

Follow these steps to build a simple shipping application and to print a test shipping label.

Build a Shipping Application

Step 1: Download the .NET framework.

Download the framework from https://www.microsoft.com/net/. **Important:** The C# SDK requires a version of the .NET framework that works with .NET Standard 1.3 or greater. For compatible frameworks, see https://docs.microsoft.com/en-us/dotnet/standard/net-standard.

Step 2: Download Visual Studio.

These instructions use Visual Studio to build the shipping application. Download Visual Studio from https://www.visualstudio.com/. Ensure Visual Studio is compatible with your version of .Net.
If you prefer a different code editor, you can use that instead.

Step 3: Create the project.

In Visual Studio (or your preferred code editor), create a new project called **MyShip**. You can choose a different name if you prefer.

Step 4: Add the NuGet package for the C# SDK.

In the **MyShip** project, add the following NuGet package as a dependency:

https://www.nuget.org/packages/shippingapi/

Step 5: Add namespaces.

In **MyShip**, open the **Program.cs** file. Include the following directives:

```
using System;
using System.IO;
using System.Text;
using System.Collections.Generic;
using PitneyBowes.Developer.ShippingApi;
using PitneyBowes.Developer.ShippingApi.Model;
using PitneyBowes.Developer.ShippingApi.Fluent;
```

Step 6: Create a session.

In the **Main()** method, create a session for connecting to the Pitney Bowes sandbox test environment:

```
static void Main()
{
 var sandbox = new Session()
 {
    EndPoint = "https://api-sandbox.pitneybowes.com",
    Requester = new ShippingApiHttpRequest()
 };
```

Step 7: Retrieve your credentials from PB Developer Hub.

a. Log into [Pitney Bowes Developer Hub].

b. Retrieve your API key and secret:

   Click **Ship** > **API Keys**. Copy your Sandbox **Key** and **Secret**.

b. Retrieve your Developer ID:

Click your **username** in the upper right corner of the page and click **Profile**. Copy your **Developer ID**.

c. Retrieve your default merchant's Shipper ID.

Click the **Ship** > **Onboarding Merchants**. Copy the Sandbox **Shipper ID**.

Step 8: Initialize the framework.

Back in **Program.cs**, in the **Main()** method, add configuration items to store the credentials you retrieved.

```
sandbox.AddConfigItem("ApiKey", "<your api key>");
sandbox.AddConfigItem("ApiSecret", "<your api secret>");
sandbox.AddConfigItem("ShipperID", "<your default merchant's shipper id>");
sandbox.AddConfigItem("DeveloperID", "<your developer id>");
```

Hook in the **Model** classes and set up the default session. If you have an existing shipping system, you can also hook in your own classes.

```
Model.RegisterSerializationTypes(sandbox.SerializationRegistry);
Globals.DefaultSession = sandbox;
```

Add the following to handle the API secret. **Important:** In a production system, encrypt the secret and hook in your decryption method here.

```
sandbox.GetAPISecret = () => new StringBuilder(configs["ApiSecret"]);
```

This finishes the setup. Next, create a shipment.

Create a Shipment

Step 1: Declare the shipment object.

In the **Main()** method, declare a **Shipment** object.

```
var shipment = ShipmentFluent<Shipment>.Create()
```

Step 2: Create the destination and origin addresses.

In the **Shipment** object, create two **Address** objects, one each for the destination and origin addresses. To validate the destination address, include the **Verify()** method.

```
.ToAddress((Address)AddressFluent<Address>.Create()
    .AddressLines("643 Greenway RD")
    .PostalCode("28607")
    .CountryCode("US")
    .Verify() // Performs address validation.
    )
.FromAddress((Address)AddressFluent<Address>.Create()
    .Company("Pitney Bowes Inc.")
    .AddressLines("27 Waterview Drive")
    .Residential(false)
    .CityTown("Shelton")
    .StateProvince("CT")
    .PostalCode("06484")
    .CountryCode("US")
    )
```

Step 3: Add the parcel dimensions.

In the **Shipment** object, create a **Parcel** object and set the parcel's weight and dimensions.

```
.Parcel((Parcel)ParcelFluent<Parcel>.Create()
    .Dimension(12, 12, 10)
    .Weight(16m, UnitOfWeight.OZ)
    )
```

Step 4: Select the label's mail class.

In the **Shipment** object, create a **Rates** object to select the mail class. This example uses USPS Priority Mail.

```
.Rates(RatesArrayFluent<Rates>.Create()
    .USPSPriority<Rates, Parameter>()
    )
```

Step 5: Configure the returned document.

In the **Shipment** object, create a **Documents** object to configure how the label is outputted. In this example, the label is returned as a URL to a PDF. The label is a 4x6 label.

```
.Documents((List<IDocument>)DocumentsArrayFluent<Document>.Create()
    .ShippingLabel(ContentType.URL, Size.DOC_4X6, FileFormat.PDF)
    )
```

Step 6: Add the Shipper ID.

In the **Shipment** object, create a **ShipmentOptions** object and retrieve the Shipper ID.

```
.ShipmentOptions(ShipmentOptionsArrayFluent<ShipmentOptions>.Create()
    .ShipperId(sandbox.GetConfigItem("ShipperID")
    )
```

Step 7: Create a unique ID for the transaction.

In the **Shipment** object, create a **TransactionId** object that uses the **Guid** class to create a unique transaction ID for every new shipment.

```
.TransactionId(Guid.NewGuid().ToString().Substring(15));
```

Step 8: Create a label.

Use the **API** class to call the **CreateShipment** method. Pass in the **Shipment** object.

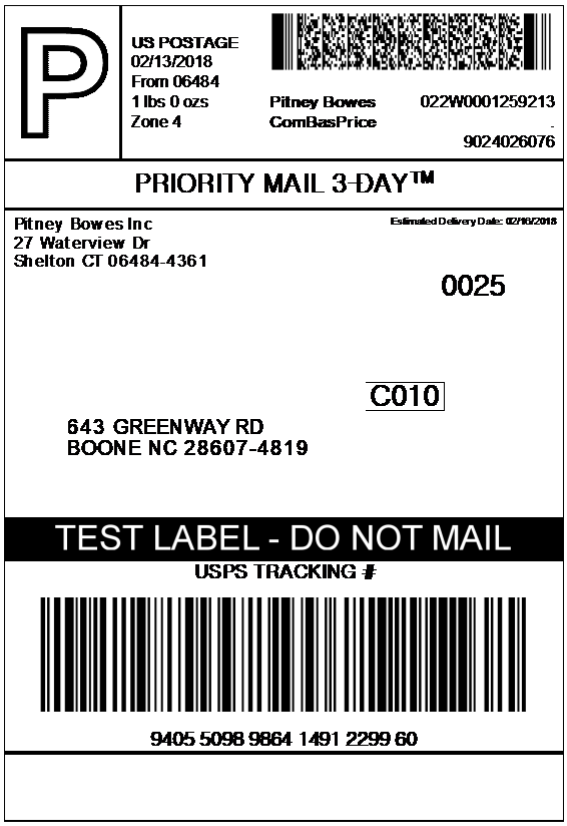
The example code below includes a **foreach** loop to handle shipments that return multiple documents (such as an international shipments, which include customs forms). The code prints each document to a separate temporary file and gives the document a temporary file name.

```
var label = Api.CreateShipment((Shipment)shipment).GetAwaiter().GetResult();
   if (label.Success)
   {
      var sw = new StreamWriter("label.pdf");
      foreach (var d in label.APIResponse.Documents)
      {
         Api.WriteToStream(d, sw.BaseStream).GetAwaiter().GetResult();
      }
   }
}}}
```

Step 9: Build and run the program to create the label.

To see the API messages exchanged with the server, run with Telerik Fiddler in background. If you receive errors, use your debugger to check for error messages in the response object.

Once the label is created, retrieve the label in your project's root directory. The following is an example label:



Questions and Issues

If you have any questions or issues, please log them on GitHub at https://github.com/PitneyBowes/pitneybowes-shipping-api-csharp.