Tutorial: Automate your SP-API Calls Using a C# SDK

This tutorial describes how to generate a C# SDK with Login with Amazon (LWA) token exchange and authentication to build your application. You can use this SDK to integrate Amazon Marketplace features into your applications, including accessing product information, managing orders, handling shipments, and more.

Prerequisites

To complete this tutorial, you need:

- A hybrid or SP-API app in draft or published state
- Integrated development environment (IDE) software (this walkthrough uses Visual Studio IDE on Windows OS)

Before your application can connect to the Selling Partner API, you must register it and have it authorized by a selling partner. If you do not have a hybrid or SP-API app, follow the steps to register as a developer, register your application, and Authorizing Selling Partner API applications. Then, return to this tutorial.

Step 1. Set up your workspace

1. On your local drive, create a directory for this project and name it `SwaggerToCL`.
2. Download the following tools:
    - IDE software (this walkthrough uses Visual Studio IDE on Windows OS)
    - GNU Wget
    - Java 8 or newer
3. Run the following command to download the Swagger Code Generator:

    Bash

    ```
    wget https://repo1.maven.org/maven2/io/swagger/swagger-codegen-cli/2.4.13/swagger-codegen-cli-2.4.13.jar -O swagger-codegen-cli.jar
    ```

4. Copy `swagger-codegen-cli.jar` into your local directory `C:\\SwaggerToCL`.
5. Use the following command to clone the `selling-partner-api-models` repository to your local directory `C:\\SwaggerToCL`.

    Bash

    ```
    git clone https://github.com/amzn/selling-partner-api-models
    ```

6. Navigate to the `selling-partner-api-models\\models` folder in your local copy of the repository and copy a JSON file from the models subfolders into `C:\\SwaggerToCL`. This tutorial uses the `sellers.json` file.
7. In GitHub, navigate to `https://github.com/amzn/selling-partner-api-`

models/tree/main/clients/sellingpartner-api-aa-csharp and download the sellingpartner-api-aa-csharp folder to your local computer.

This folder provides helper classes to generate an access token for the Amazon Selling Partner APIs. The helper classes are intended for use with the Selling Partner API Client Libraries generated by Swagger Codegen using the RestSharp library.

The next step is to generate the C# SDK with the authentication and authorization classes provided in the sellingpartner-api-aa-csharp folder.

Step 2. Generate a C# SDK with LWA token exchange and authentication

1. Open Visual Studio.
2. In the **sellingpartner-api-aa-csharp** folder, select the `SellingPartnerAPIAuthAndAuthCSharp.sln` file. Select **build** in Visual Studio, which generates the `Amazon.SellingPartnerAPIAA.dll` assembly in the folder **sellingpartner-api-aa-csharp\src\Amazon.SellingPartnerAPIAA\bin\Debug\netstandard2.0**.
3. Open a terminal and run the following commands to generate the C# client library. The default package name for the generated client library is `Swagger.IO`. The commands generate client libraries with their respective API names as the package name, rather than `Swagger.IO`.
    - In `C:\\SwaggerToCL`, create a JSON file named `csharpConfig.json`. Open an editor and add the following code. For `packageName`, use the same name of the API you want to generate the client library for:

    JSON

    ```
    {"packageName":"SellingPartnerAPI.SellerAPI","targetFramework":"v4.7.2"}
    ```

    Run this command to generate C# code with a customized package name:

    Bash

    ```
    java -jar C:\SwaggerToCL\swagger-codegen-cli.jar generate -i C:\SwaggerToCL\[name of model].json -l csharp -t [path to selling-partner-api-models\clients\sellingpartner-api-aa-csharp folder]\src\Amazon.SellingPartnerAPIAA\resources\swagger-codegen\templates\ -o C:\SwaggerToCL\[name of client library] -c C:\SwaggerToCL\csharpConfig.json
    ```

    This command uses `Sellers.json` to generate C# code:

    Bash

    ```
    java -jar C:\SwaggerToCL\swagger-codegen-cli.jar generate -i C:\SwaggerToCL\Sellers.json -l csharp -t C:\SwaggerToCL\sellingpartner-api-aa-csharp\src\Amazon.SellingPartnerAPIAA\resources\swagger-codegen\templates\ -o C:\SwaggerToCL\Sellers_CsharpCL -c C:\SwaggerToCL\csharpConfig.json
    ```

    The SDK is created in `C:\\SwaggerToCL\\Sellers_CsharpCL`. The next step is to write the actual application and connect to the SP-API using the generated C# SDK.

Step 3. Connect to the Selling Partner API using the generated C# SDK

1. In Visual Studio, navigate to your generated library **Sellers_CsharpCL** folder and open the `.sln` file. The `.sln` package file will have the name as provided in Step 2. Generate a C# SDK with LWA token exchange and authentication. In this example, the package file name is `SellingPartnerAPI.SellerAPI.sln`.
2. Right-click the `SellingPartnerAPI.SellerAPI` file and select **Add > References**.
3. In the **References** window, select **Browse**. Navigate to `sellingpartner-api-aa csharp/src/Amazon.SellingPartnerAPIAA/bin/Debug/netstandard2.0` and select `Amazon.SellingPartnerAPIAA.dll`.
4. Repeat steps 1-3 for the `SellingPartnerAPI.SellerAPI.Test` file.
5. Navigate to your generated library `Sellers_CsharpCL` folder. Right-click the `SellingPartnerAPI.SellerAPI` file, choose **Options > General**, and change the target framework to **.net Framework v4.7 and newer**.
6. Repeat step 5 for the `SellingPartnerAPI.SellerAPI.Test` file.
7. Right-click the `SellingPartnerAPI.SellerAPI` file and select **Manage NuGet Packages**. Ensure you have the following package versions (If not, install the respective NuGet packages):
    - `JsonSubTypes 1.2.0` or newer
    - `Newtonsoft.json 12.0.3` or newer
    - `RestSharp 106.12.0`
    - `RateLimiter 2.2.0` or newer (this will also install `ComposableAsync.Core`)
8. Right-click the `SellingPartnerAPI.SellerAPI.Test` file and select **Manage NuGet Packages**. Install the **Nunit 2.6.4** package.
9. In the `SellingPartnerAPI.SellerAPI.Test > Api` folder, open the `[Modelname]Tests.cs` file and add the following code:

    C#

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Reflection;
using RestSharp;
using SellingPartnerAPI.SellerAPI.Client;
using SellingPartnerAPI.SellerAPI.Api;
using SellingPartnerAPI.SellerAPI.Model;
using Amazon.SellingPartnerAPIAA;

namespace SellingPartnerAPI.SellerAPI.Test
{
    class SellersApiTests
    {
```

```
        public static void Main(string[] args)
        {
            try
            {
                LWAAuthorizationCredentials lwaAuthorizationCredentials = new
LWAAuthorizationCredentials
                {
                    ClientId = "amzn1.application-oa2-client.******************",
                    ClientSecret = "**********************************",
                    RefreshToken = "Atzr|**********************************",
                    Endpoint = new Uri("https://api.amazon.com/auth/o2/token")
                };
                SellersApi sellersApi = new SellersApi.Builder()
                    .SetLWAAuthorizationCredentials(lwaAuthorizationCredentials)
                    .Build();

                GetMarketplaceParticipationsResponse result =
sellersApi.GetMarketplaceParticipations();
                Console.WriteLine(result.ToJson());
            }
            catch (LWAException e)
            {
                Console.WriteLine("LWA Exception when calling
SellersApi#getMarketplaceParticipations");
                Console.WriteLine(e.getErrorCode());
                Console.WriteLine(e.getErrorMessage());
                Console.WriteLine(e.Message);
            }
            catch (ApiException e)
            {
                Console.WriteLine("Exception when calling
SellersApi#getMarketplaceParticipations");
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

`LWAAuthorizationCredentials` instance parameters:

Name	Description	Required
`ClientId`	Your LWA client identifier. For more information, refer to Viewing your application information and credentials.	Yes
`ClientSecret`	Your LWA client secret. For more information, refer to Viewing your application information and credentials	Yes

Name	Description	Required
`RefreshToken`	The LWA refresh token. Get this value when the selling partner authorizes your application. For more information, refer to Authorizing Selling Partner API Applications.	No. Include `RefreshToken` if the operation that you call in the following step requires selling partner authorization. All operations that are not grantless operations require selling partner authorization. If you include `RefreshToken`, do not include `Scopes`.
`Scopes`	The scope of the LWA authorization grant. You can specify one or more `Scopes` values: `ScopeNotificationsAPI`. For the Notifications API. - `ScopeMigrationAPI`.	No. Include `Scopes` if the operation that you call in the following step is a grantless operation. If you include `Scopes`, do not include the `RefreshToken`.
`Endpoint`	The LWA authentication server URI.	Yes

Example of an operation call that requires selling partner authorization:

```
using Amazon.SellingPartnerAPIAA;
LWAAuthorizationCredentials lwaAuthorizationCredentials = new LWAAuthorizationCredentials
{
  ClientId = "myClientId",
  ClientSecret = "myClientSecret",
  RefreshToken = "Aztr|...",
  Endpoint = new Uri(""https://api.amazon.com/auth/o2/token"")
};
```

Example of a grantless operation call:

```
using Amazon.SellingPartnerAPIAA;
LWAAuthorizationCredentials lwaAuthorizationCredentials = new LWAAuthorizationCredentials
{
  ClientId = "myClientId",
  ClientSecret = "myClientSecret",
  Scopes = new List<string>() { ScopeConstants.ScopeNotificationsAPI,
ScopeConstants.ScopeMigrationAPI }
  Endpoint = new Uri(""https://api.amazon.com/auth/o2/token"")
};
```

10. To view the output of the code, you must convert it to a console application. To do so, right-click the `SellingPartnerAPI.SellerAPI.Test` file and select **Options**.
11. In the **Project Options** window, in the left navigation pane, expand **Build**, then select **General**. In the **Code Generation** section, select **Compile Target** and select **Executable with GUI**. Select **OK**.
12. Build and run the project.