AppMill

CUSTOMER DLL PROJECT CREATION AND USAGE IN APPMILL SERVICE

Internal Technical Documentation

Table of Contents

1 Introduction

The Customer DLL Project Creation and Usage Example in AppMill Service guide contains detailed instructions and guidelines on how to create, integrate, and utilize a custom DLL (Dynamic Link Library) within the AppMill service environment. It serves as a reference for developers who want to extend the functionality of their AppMill service by leveraging external libraries.

Using custom DLLs (Dynamic Link Libraries) in AppMill services offers several advantages and serves various purposes, enhancing the functionality and flexibility of the services.

2 Customer DLL Project Creation

To create and use a customer DLL (Dynamic Link Library) in the AppMill service, follow these general steps:

- Create the DLL Project:
  - Use a development environment like Visual Studio to create a new Class Library project.
  - Write the code for custom functionality, such as helper methods, data processing functions, or any other reusable components.
- Build the DLL Project:
  - Once the code is written, build the project to generate the DLL file.
  - Make sure to set the build configuration to target the appropriate architecture (e.g., x86, x64) based on your requirements.
- Reference the DLL Project in AppMill:
  - In the AppMill project, add a reference to the DLL created.
  - This typically involves locating the DLL file and adding it as a reference in your AppMill project settings.
- Use the DLL Project in AppMill:
  - Once the DLL is referenced, use its functionality within the AppMill service.
  - This might involve instantiating objects from classes defined in the DLL, calling methods, or accessing properties as needed.

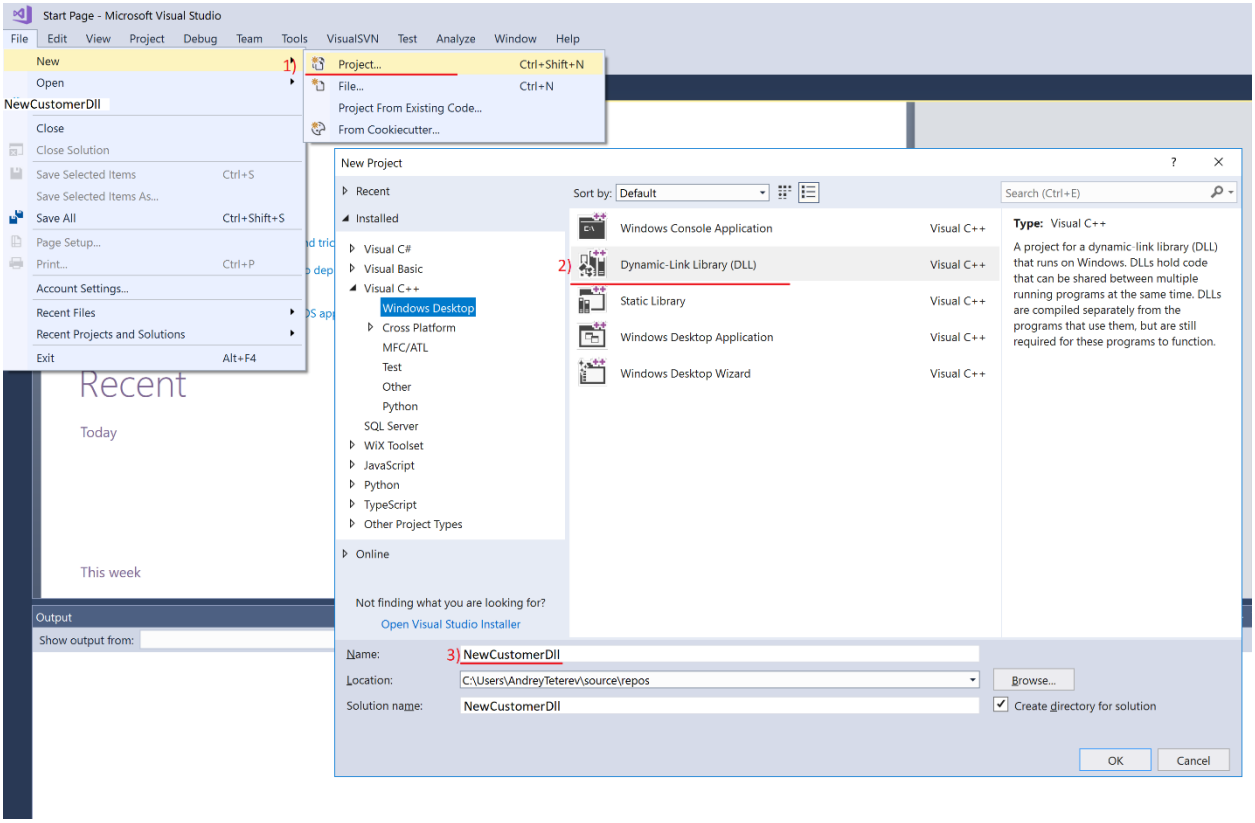
As the example in that directory has been stored the zip archive with the example of a DLL project that emulate an empty customer DLL.

The Customer API consist of:

- **AppMillSDK.API.h** Header file – contains all necessary classes and functions definitions to use the AliasFramework objects and additional helpers.
- **AppMillSDK.AP.h** Library – contains exported methods and classes.

Perform the following steps to create a dll that will be used as a module in the AppMill service:

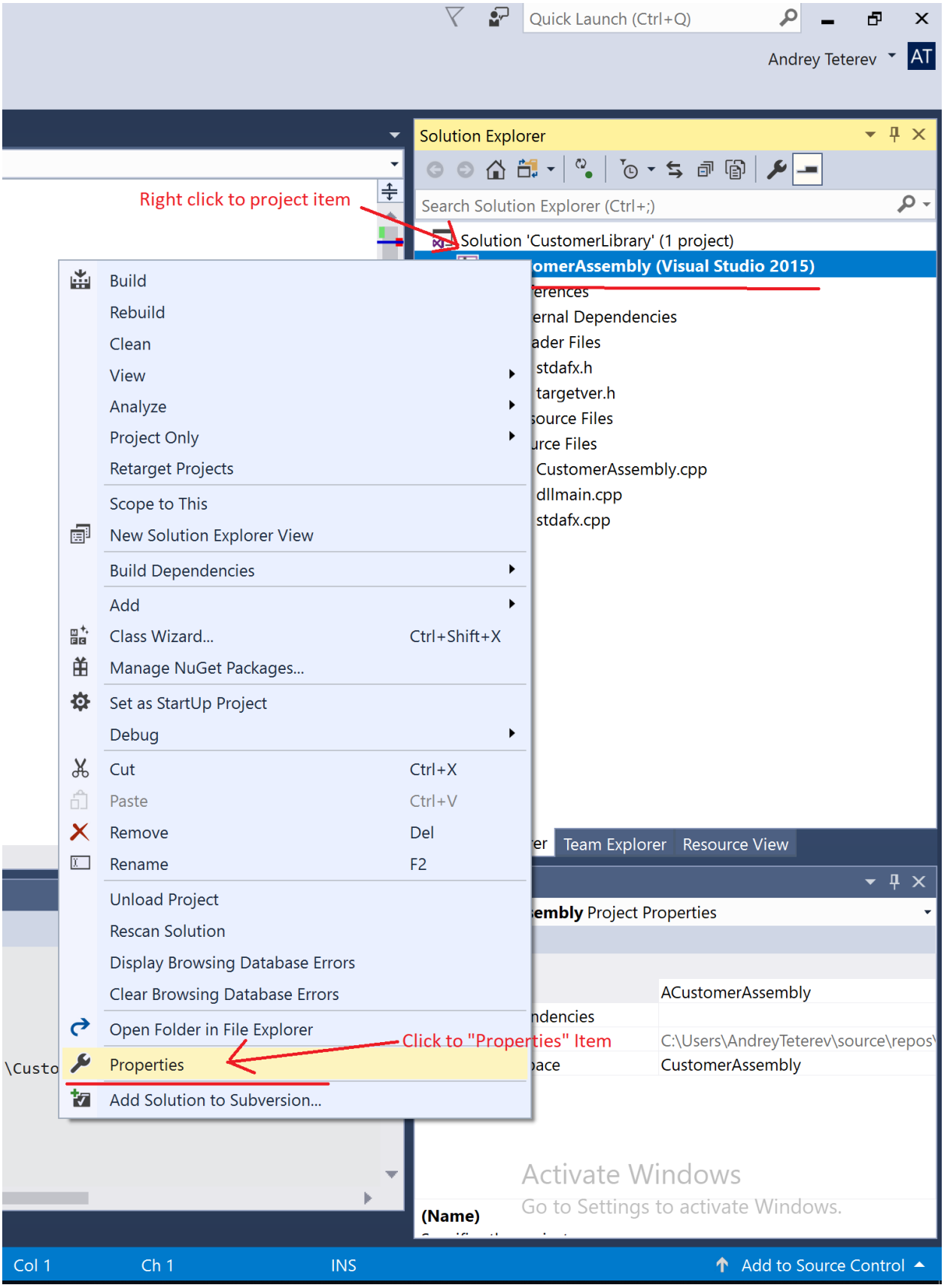
1. Create an empty dll project in the Visual Studio



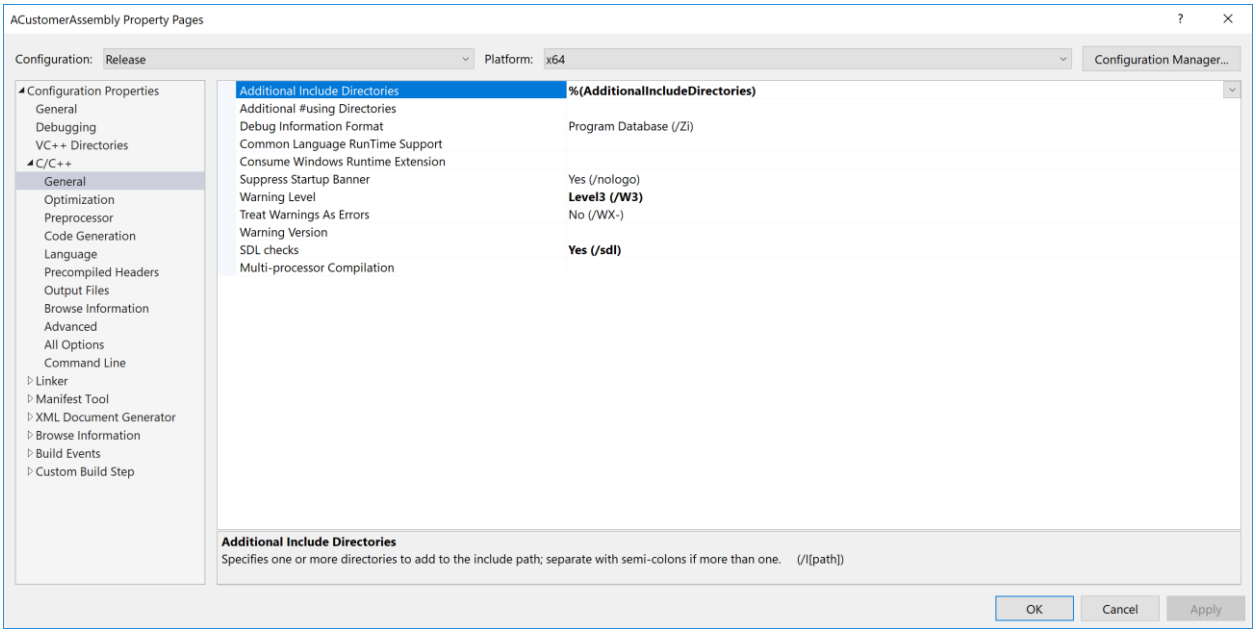
To create an empty dll project in the Visual Studio, in the **File** menu:

a. click the **New** menu item and then select the Project… option. The New Project pop-up window opens.
b. in the New Project pop-up window, select the Dynamic-Link Library (DLL).
c. enter a new dll project name. Click the OK button to finish the dll project creation.

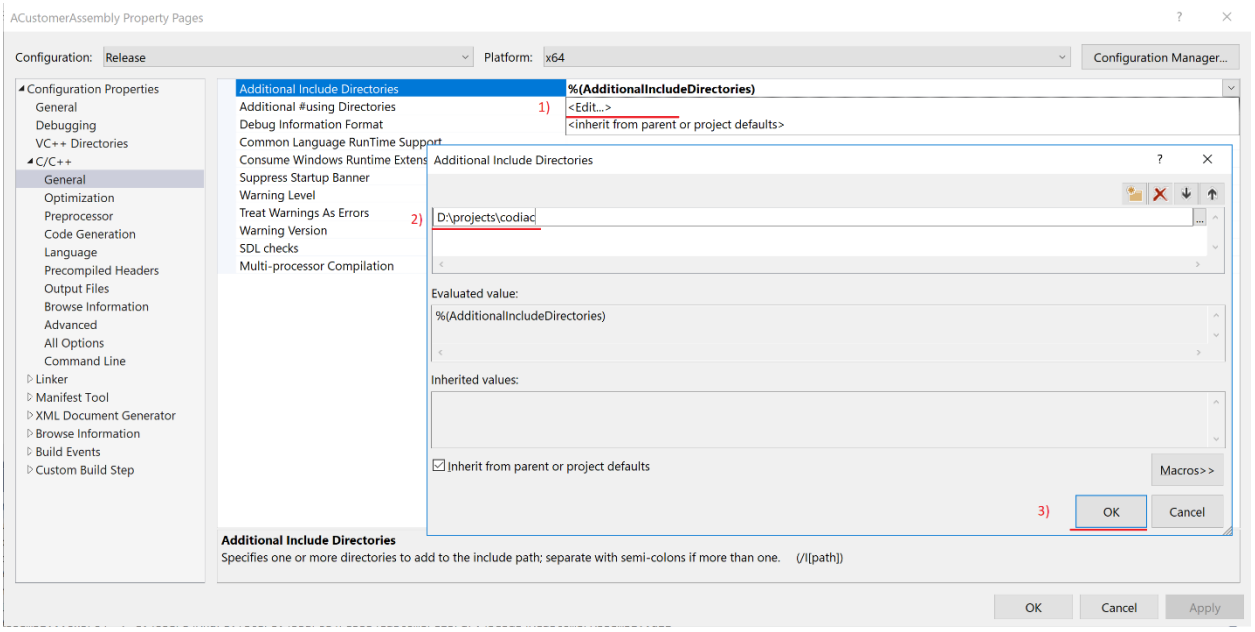
2. In the following step, you need to change the project settings to add include directories to the header file, a path to the library, preprocessors definitions. For that:
   a. right click on the project item and
   b. select the **Properties** menu item.



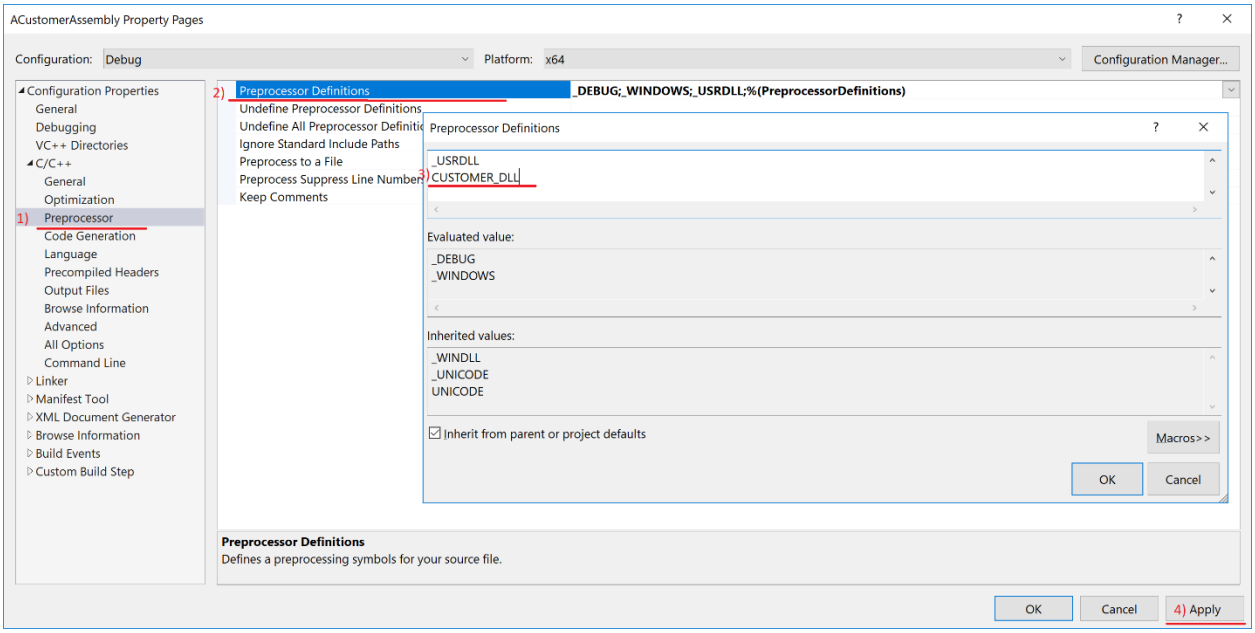
The following window opens.



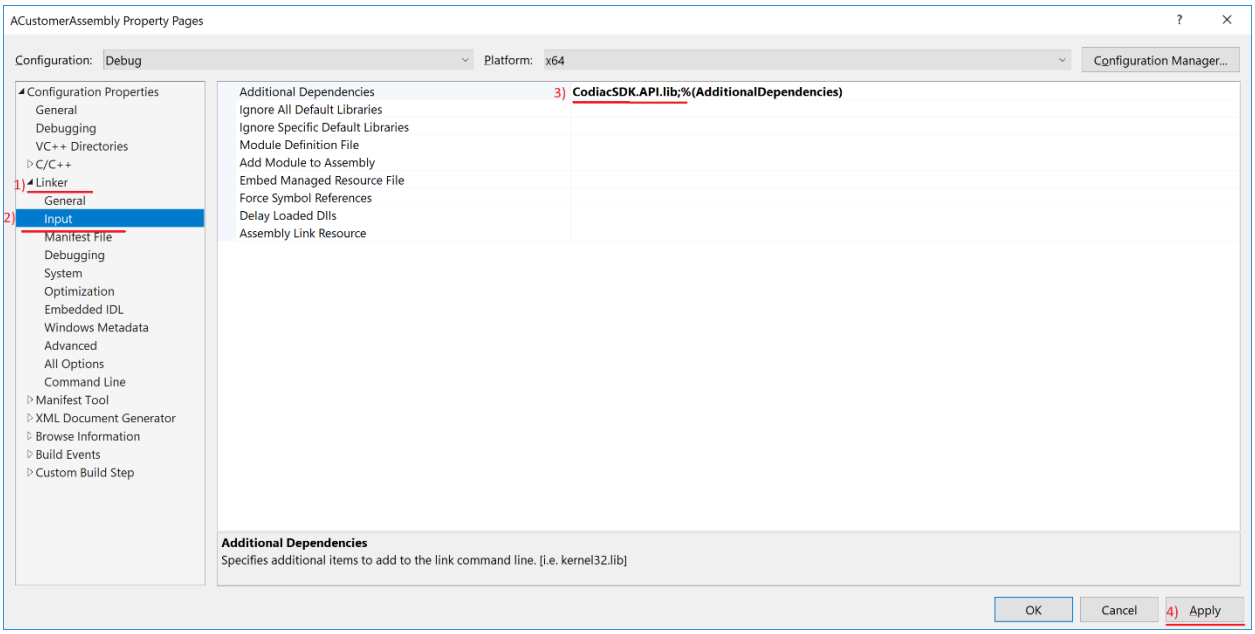
3. Add path to the place where the **AppMillSDK.API.h** file has been stored, for example D:\projects\AppMill. For that perform the following steps:
    a. select in the left tree **C\C++** > **General** and click **<Edit…>**.
    b. enter the appropriate path and click the **OK** button.



4. Add the Preprocessor definition **CUSTOMER_DLL** to export function that marks dll as a AppMill module. For that:
    a. click the Preprocessor menu item.
    b. select the Preprocessor Definitions option.
    c. in the opened Preprocessor Definitions, add the preprocessor definition and click the **OK** button.
    d. click the **Apply** button to finish the process.



5. Add a dependency from the **AppMillSDK.API.lib** to the Linker settings. For that:
    a. click the Linker menu item.
    b. click the Input menu item.
    c. select the **AppMillSDK.API.lib;%(AdditionalDependencies)** option.
    d. click the **Apply** button to finish the process.



6. Create a new or open an existing cpp file and add a function that has the following signature (also, we need to add include directives for AppMillSDK.API.h):

```
#include "AppMillSDK.API.h"

wchar_t * CustomerFunction(ObjectPtr<IAliasFramework> af, ObjectPtr<IAliasValuesMap> avm)

{

	//TODO: Add here your code

	return _AFResponce(af, avm);
```

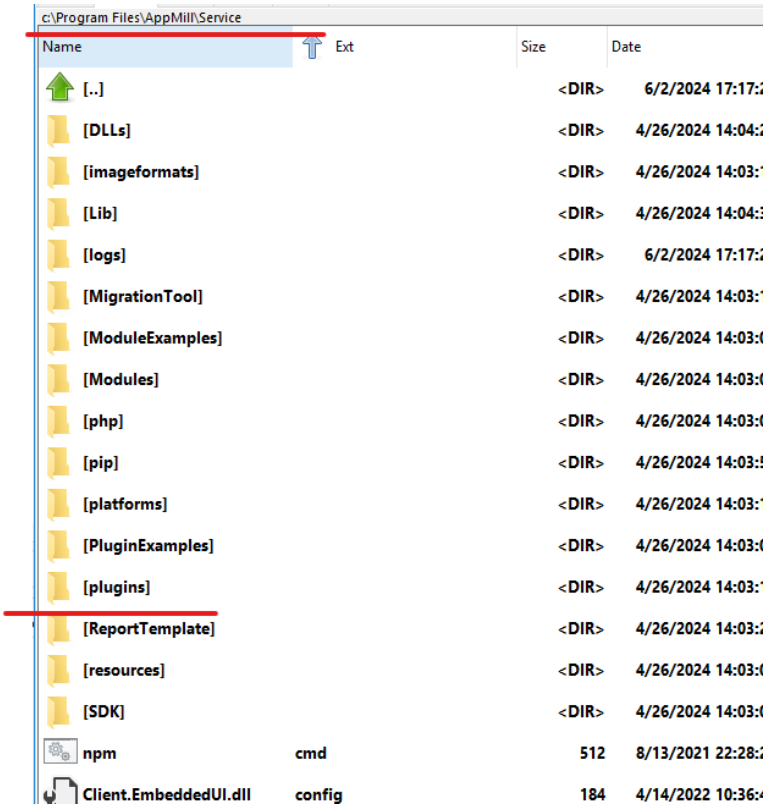
```
}

REGISTER_CALLBACK(CustomerFunction);
```

where:

- **ObjectPtr** – a class that implements the logic that controls the life cycle of the exported interfaces (base implantation of the smart pointers).
- **_AFResponce** – a helper function that constructs a valid response for the AppMill core.
- **REGISTER_CALLBACK** – macros that performs a customer function registration in the AppMill core.

7. Build the project and put the result dll file into the root folder of the AppMill Service: **AppMill/Service/plugins**.

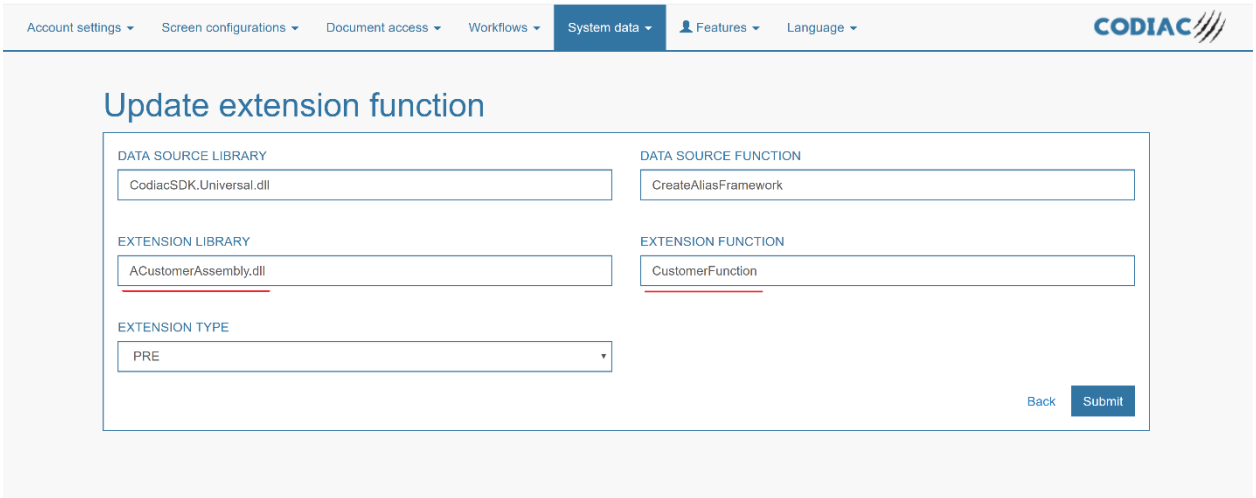


3 Customer DLL Project Usage

The Customer DLL is a mechanism designed to extend the existing service with customized functionality tailored to meet specific customer requirements. This allows for the integration of additional processing capabilities and business processes beyond the standard functionality provided.

Customers can create their own plugins, enabling them to customize screens, threads, and other components of the Service to better align with their unique business needs.

For example, the created customer DLL project can be used to create new Extension function in the Architect application.



To create new Extension function in the Architect application, perform the following steps:

- Open the **Extension function** functionality (System data > Extension functions)
- Click the **Create** button, to create a new extension function. This opens the page in creation mode.
- Fill in the following fields:
    - Data Source Library – select the data source library. In this example, AppMillSDK.Universal.dll.
    - Data Source Function – enter a name for a new extension function: CreateAliasFramework.
    - Extension Library – select the extension library. In this example, ACustomerAssembly.dll.
    - Extension Function – enter a class name and function name of the extension function. For example, CustomerFunction.
    - Extension Type – select the extension type from the drop-down list. In our case, PRE.
- Click the **Submit** button to save the entered data.

After creating the extension function, it can be used as a PRE method for creating new items.