AppMill

PLUGINS AND ALIAS FRAMEWORK EXTENSION FUNCTIONS

Revisions

Revision	Date	Description
0.1	03/25/21	Initial revision: • added documentation for C++ plugins and AF ext functions. • added documentation for Python plugins and AF ext functions.
0.2	03/27/21	• added documentation for PHP plugins and AF ext functions. • added documentation for JavaScript plugins and AF ext functions.
0.3	03/29/21	• added architecture section. • added configuration of custom alias. • added usage of native database library.
0.4	03/30/21	• added formatting, rephrase some of the paragraphs, corrected typo errors.
0.5	05/12/21	• added mark for deletion API and is mark for deletion API. • added documentation for C# plugins and AF ext functions. • added examples for alias setup and plugin setup.
0.6	05/28/21	• added npm package install. • added PHP Oracle SQL. • added php.ini.
0.7	07/01/21	• added build links and C++ module configuration for C++ database libraries.
0.8	10/13/21	• added Pavel's changes to section 3.1, 3.3, 11.3, 12.4, 15.3, 16.4. • added Alex D's changes to section 5.2. • formatting
0.9	04/25/23	• added Document Management plugin functions (sections: 6.7, 9.6, 13.6, 17.6, 21.6). • added SendEmail function with attachments (sections: 6.9.2, 9.8.2, 13.8.2, 17.8.2, 21.8.2).
1.0	04/03/24	• added remote debugging instructions for C++ Python, JavaScript, PHP, and C#. • added details about Python package manager. • added details about PHP additional libraries and package manager.

1.1	04/18/24	• added details about JavaScript package manager.

Table of Contents

1 Introduction

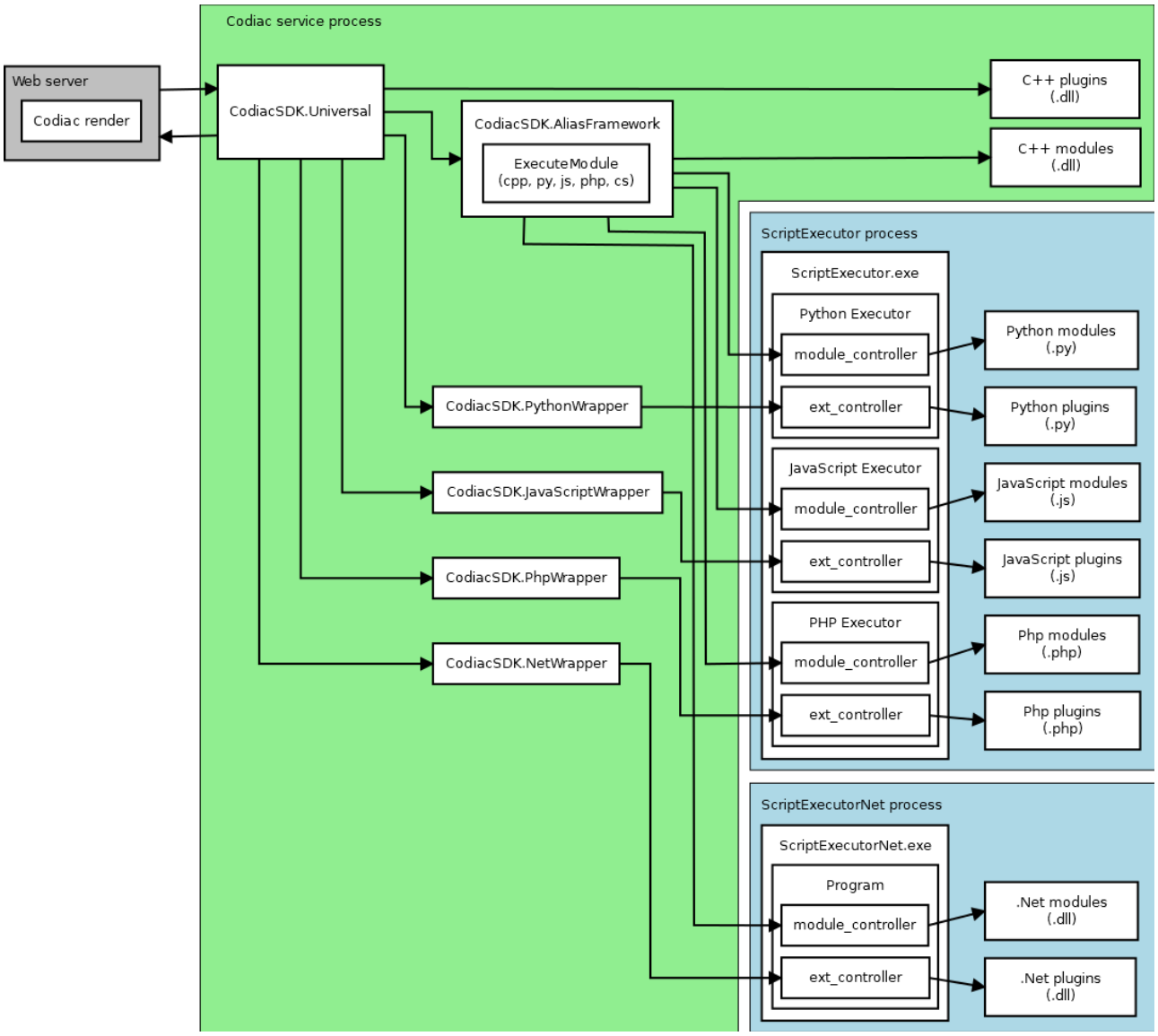
This document covers the high-level design, setup and usage of extension functions (or plugins) and Alias Framework (AF) extension functions.

The extension functions are executed by Codiac service and can be assigned as pre-action or post-action to client operations, as an execute function, or as a scheduled job.

The AF extension functions are executed by AF when custom aliases are evaluated. These functions can perform tasks or make calculations and return evaluated aliases.

2 Architecture

Below is the high-level architecture showing the relations of Codiac service to AF extension modules and plugins.



- Codiac render is a web application that send request to **`CodiacSDK.Universal`** service.
- Codiac service process contain many services. These are the services for plugins and AF extension modules:

  a) **`CodiacSDK.PythonWrapper`** service gets the list of Python plugins and spawns **`ScriptExecutor`** process with **`ext_controller`** parameter to execute Python plugin function.

  - **`ScriptExecutor`** process loads and executes Python scripts.
  - The python script imports pyd file which provides Codiac SDK library and Alias Framework library.

  a) **`CodiacSDK.JavaScriptWrapper`** service gets the list of JavaScript plugins and spawns

ScriptExecutor process with **ext_controller** parameter to execute JavaScript plugin function.

- **ScriptExecutor** process loads and executes JavaScript scripts.
- V8 wrapper provides APIs to Codiac SDK library and Alias Framework library to JavaScript scripts.

a) **CodiacSDK.PhpWrapper** service gets the list of PHP plugins and spawns **ScriptExecutor** process with **ext_controller** parameter to execute plugin function.

- **ScriptExecutor** process uses PHP library to load and execute PHP scripts.
- The PHP script loads Zend extension which provides APIs to Codiac SDK library and Alias Framework library.

a) **CodiacSDK.NetWrapper** service get the list of C# plugins (function names of C# dll files) and spawns **ScriptExecutorNet** process with **ext_controller** parameter to execute a C# plugin.

a) **CodiacSDK.AliasFramework** service (for Python/JavaScript/PHP AF extension function) spawns **ScriptExecutor** process with **module_controller** parameter to execute AF extension function.

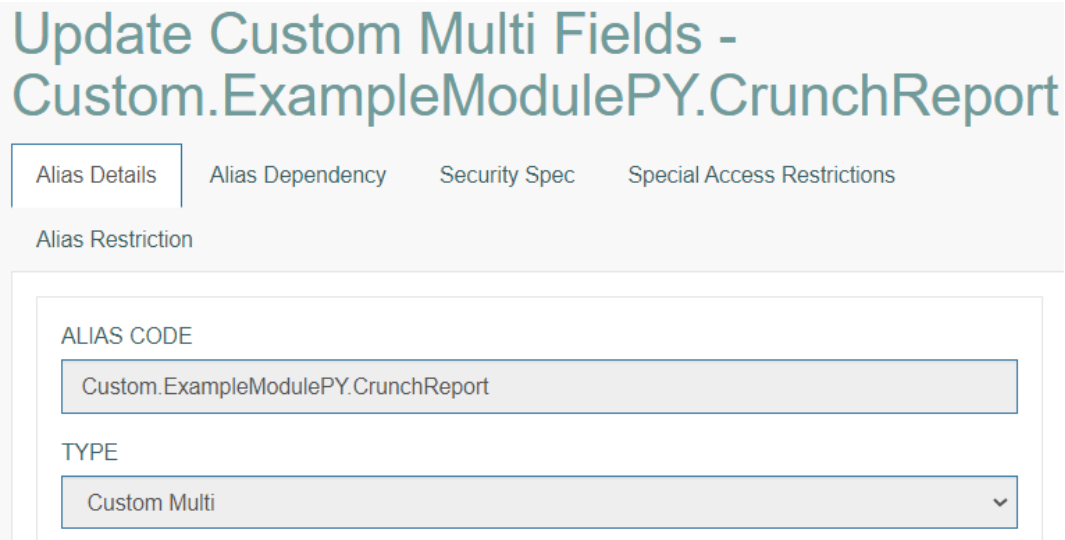
a) **CodiacSDK.AliasFramework** service (for C# AF extension function) spawns **ScriptExecutorNet** process with **module_controller** parameter to execute a C# AF extension function.

- C++ plugins are directly executed by **CodiacSDK.Universal** service and have access to Codiac SDK library and Alias Framework library.
- C++ modules are directly executed by **CodiacSDK.AliasFramework** service during evaluation of custom aliases. These modules have access to CodiacSDK library and AliasFramework aliases.

3 Configuration of Custom Alias

Below is the configuration of custom alias in PHP client. In Codiac builder, this is in **Alias Management → Custom Multi** for custom multi alias; and **Alias Management → Custom Generated** for custom generated alias.

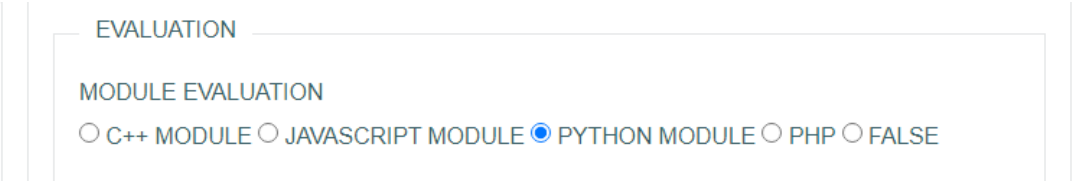
The top part of configuration has the alias code and alias type.



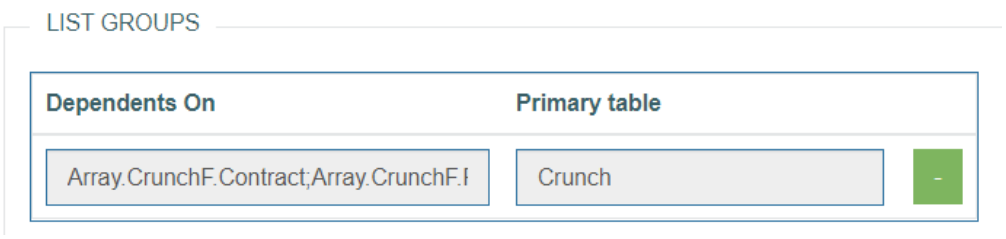
The custom alias code format follows the *AliasType,ModuleName.FunctionName* format.

- The *AliasType* can be **Custom Generated (**for Custom Generated type) if the custom alias have only one value or **Custom Multi** (for Custom Multi type) if the custom alias have multiple values (array).
- The *ModuleName* refers to class name.
- The *FunctionName* refers to function name.

The bottom part has the module evaluation which refers to module language of the custom alias. So depending on the selected module, AF will invoke the specific language interpreter for Python/PHP/JavaScript to execute a script; or execute the C++ or C# module.



The Alias Dependency setting (second tab) has the aliases that can be access by Alias Framework (AF) extension module. This means that the module functions can get and set the values of dependent aliases using provided API of Alias Framework.

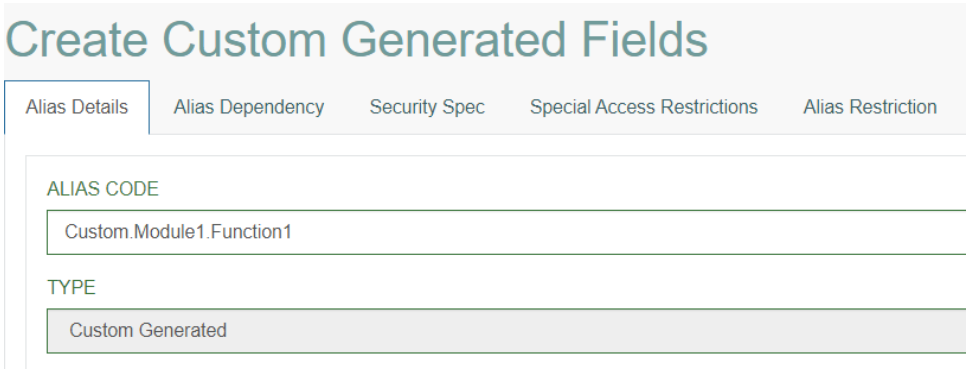


See the Alias Framework (AF) extension module sections on how the AF handles custom alias for different language types.

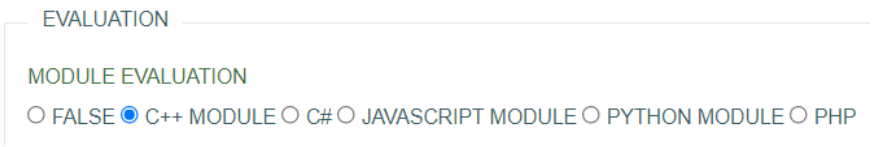
3.1 How to setup a custom generated alias?

Let say, we have created a C++ module (`MyModule.dll`) that has `Function1` method.

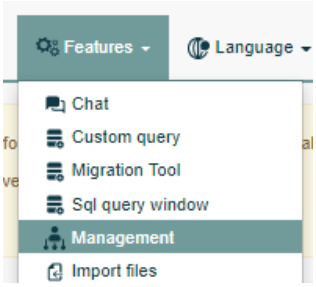
1. Go to **Alias Management → Custom Generated → Create**
2. Put **Custom.Module1.Function1** in ALIAS CODE as shown below.



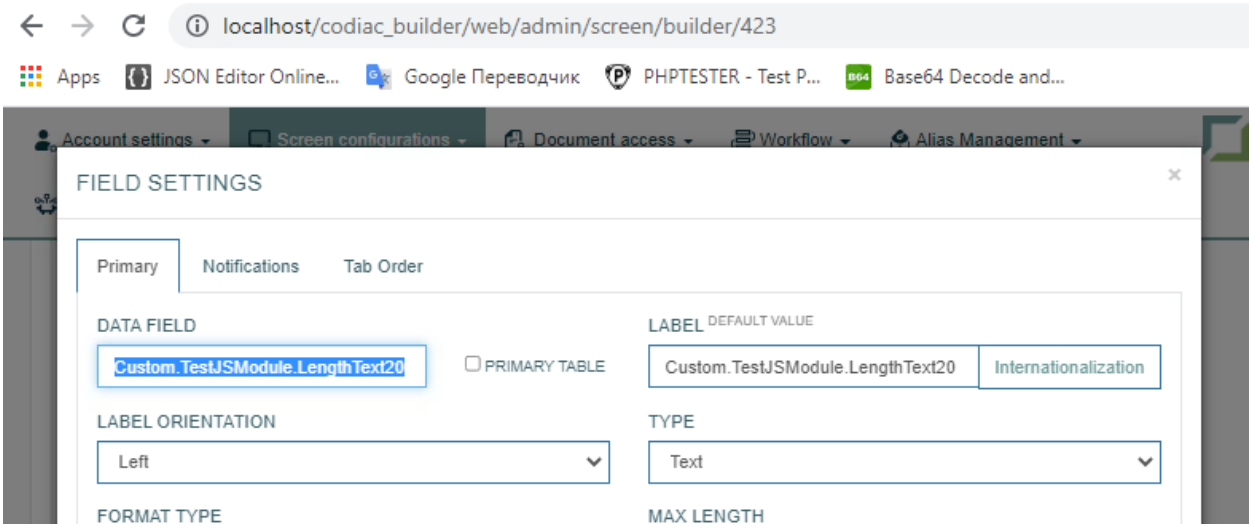
3. Select **C++ MODULE** in MODULE EVALUATION section; if not a C++ module, select appropriate language module



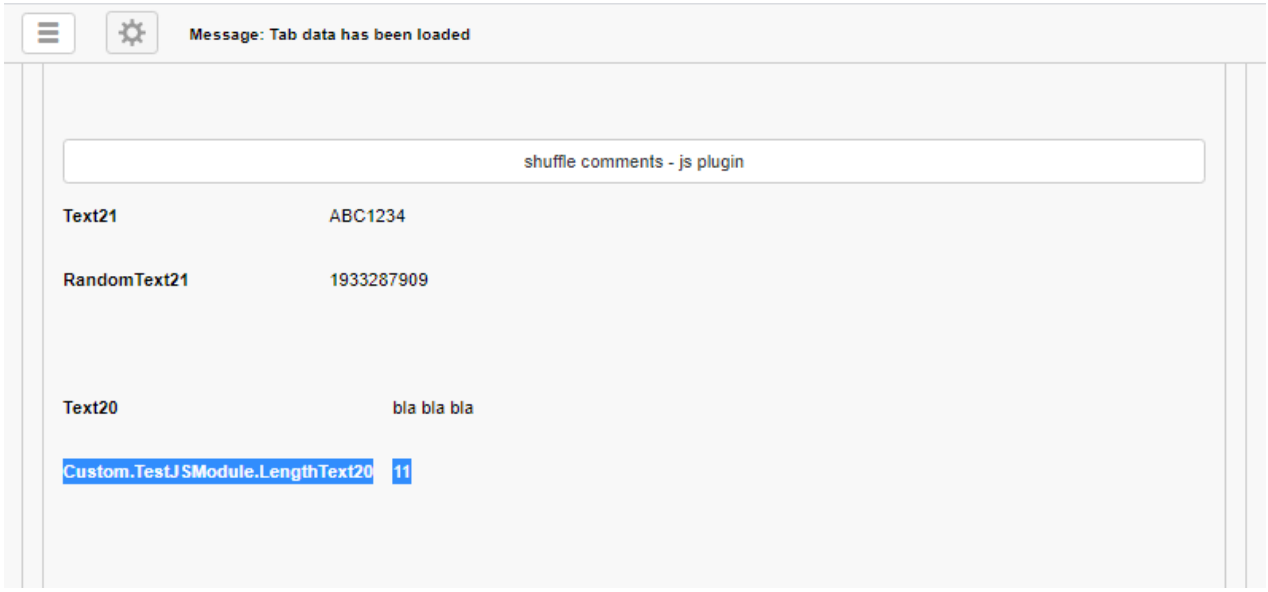
4. Click **Submit** to save the alias setup
5. Go to the page **Features → Management** and click **Reload aliases** button



6. Use an alias in the screen designer for field setting



7. When loading a screen containing a module alias, the function bound to the alias will be executed.



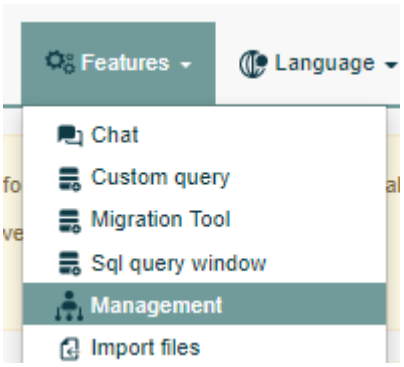
3.2 How to setup a custom multi alias?

The steps is similar to setting up a custom generated alias. Let say, we have the same C++ module (`MyModule.dll`) that has the same `Module1` class but with `Function2` method.

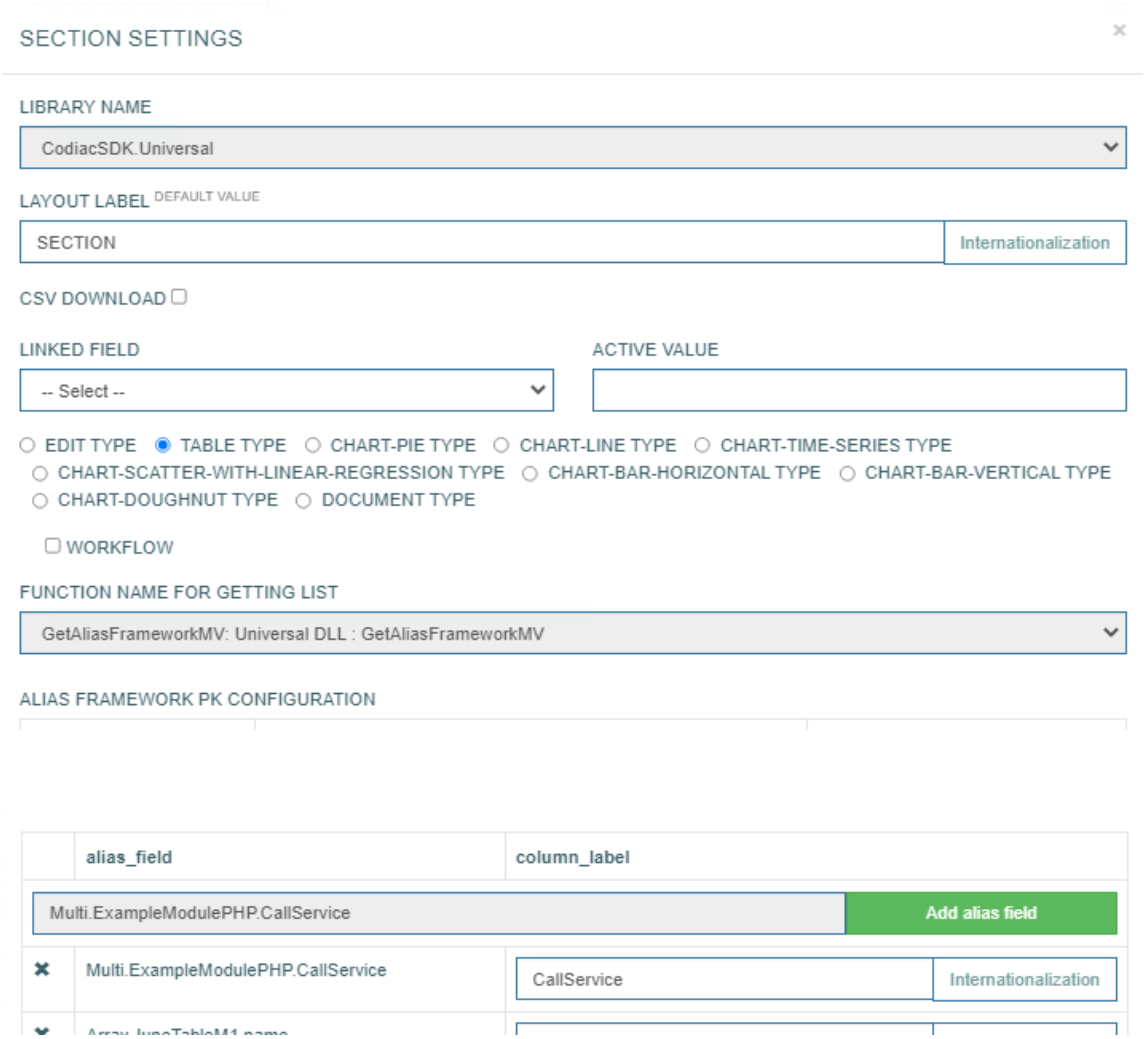
1. Go to **Alias Management → Custom Multi → Create**
2. Put **Multi.Module1.Function2** in ALIAS CODE as shown below.

3. Select **C++ MODULE** in MODULE EVALUATION section; if not a C++ module, select appropriate language module

4. Click **Submit** to save the alias setup
5. Go to the page **Features** → **Management** and click **Reload aliases** button



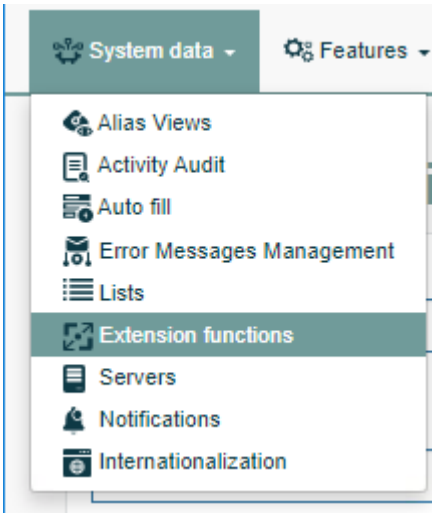
6. Use an alias in the screen builder to customize the table or chart field (where **`GetAliasFrameworkMV`** is used)



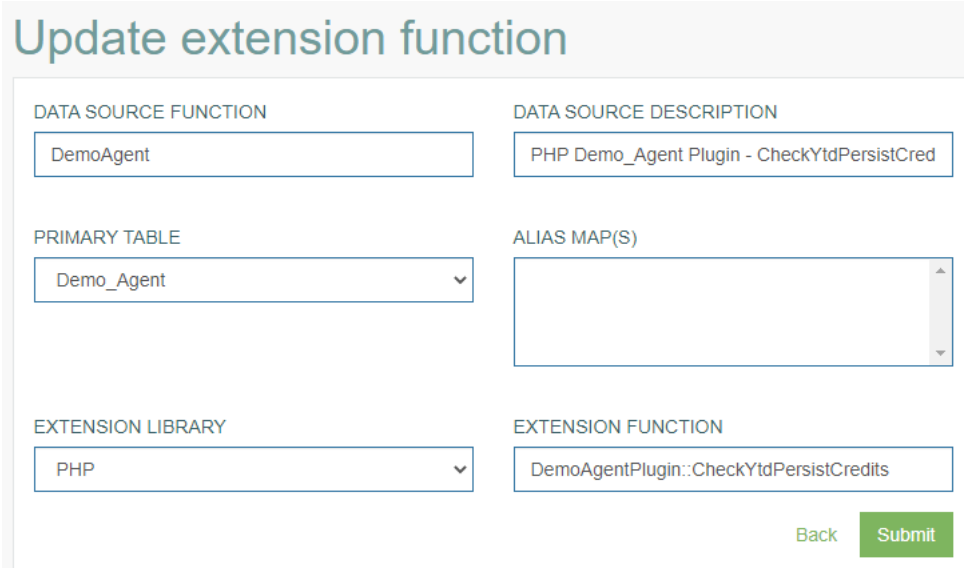
7. When loading a screen containing a module alias, the function bound to the alias will be executed.

3.3 How to use a plugin?

1. Create a plugin file as indicated in this instruction
2. Restart the service
3. Add your plugin functions in Codiac builder
   a) Add your plugin functions to extension function setup

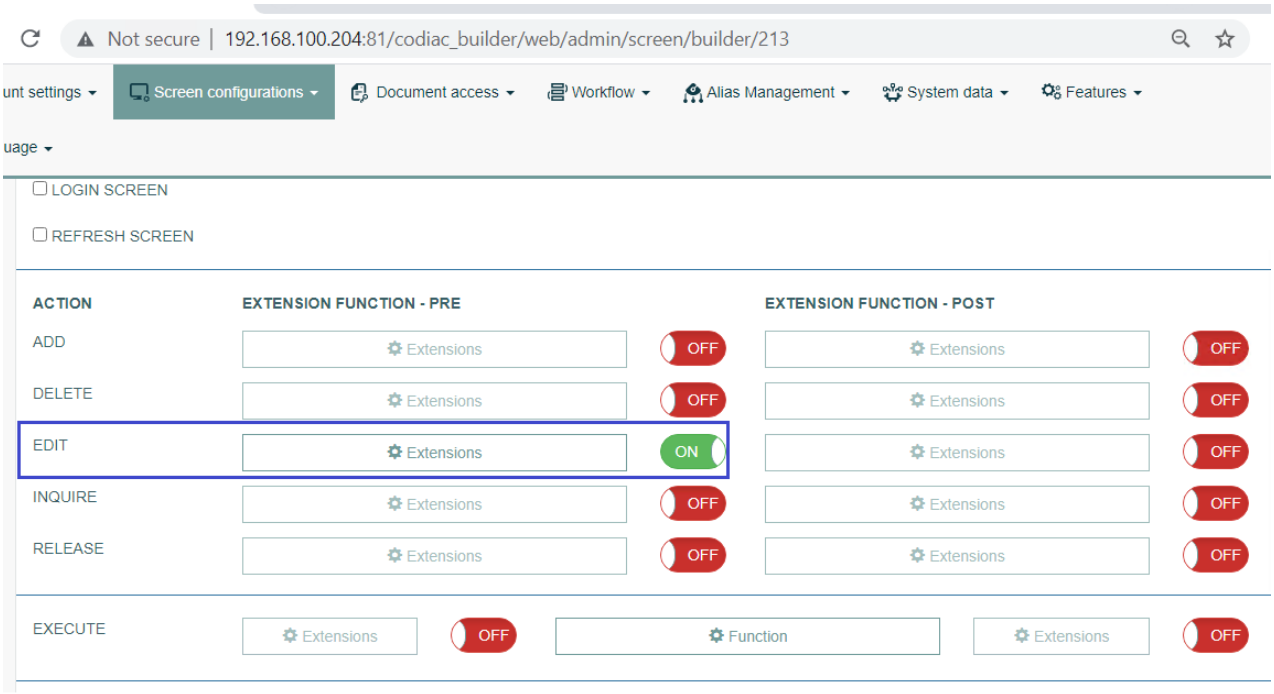


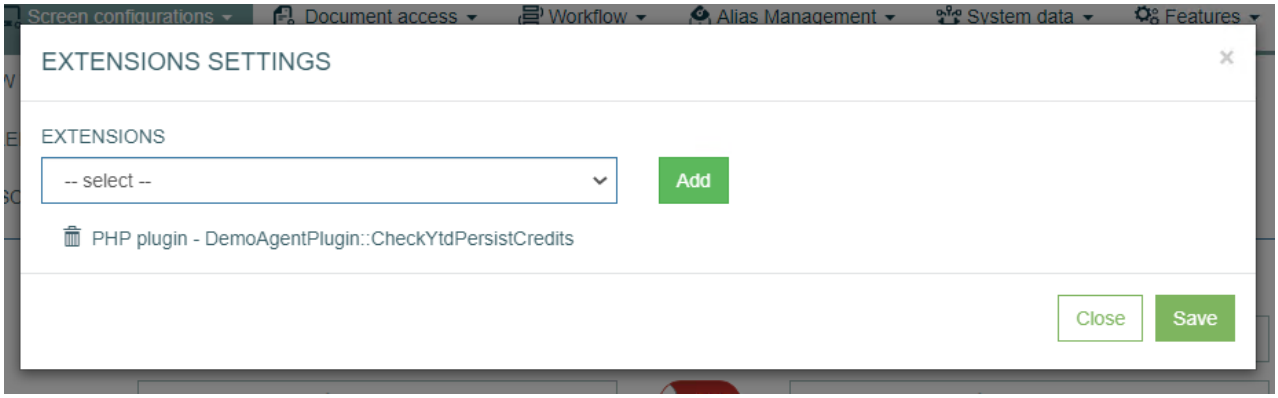
The language in which the plugin is written is selected in the **Extension Library**, also you may indicate it in other places (but it isn't require).



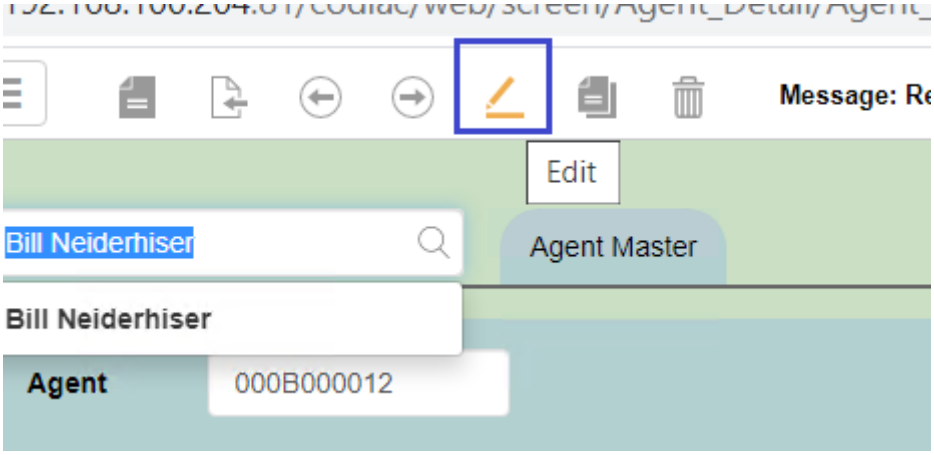
- **Data source function** – This field can be any name and is only use for UI filtering.
- **Primary table** – The table that plugin will be attached to. This will show up in extension function list of screen builder.
- **Extension library** – Set to script language of the script. For example, if plugin function is a PHP script then set to PHP.
- **Data source description** – This field can be any name. It is only use as a name in extension function list of screen builder.
- **Alias map** – The Alias View data that plugin will receive in JSON parameter.
- **Extension function** – The class name and function name (separated by double semicolon) to execute.

b) Configure your screen to include the extension functions

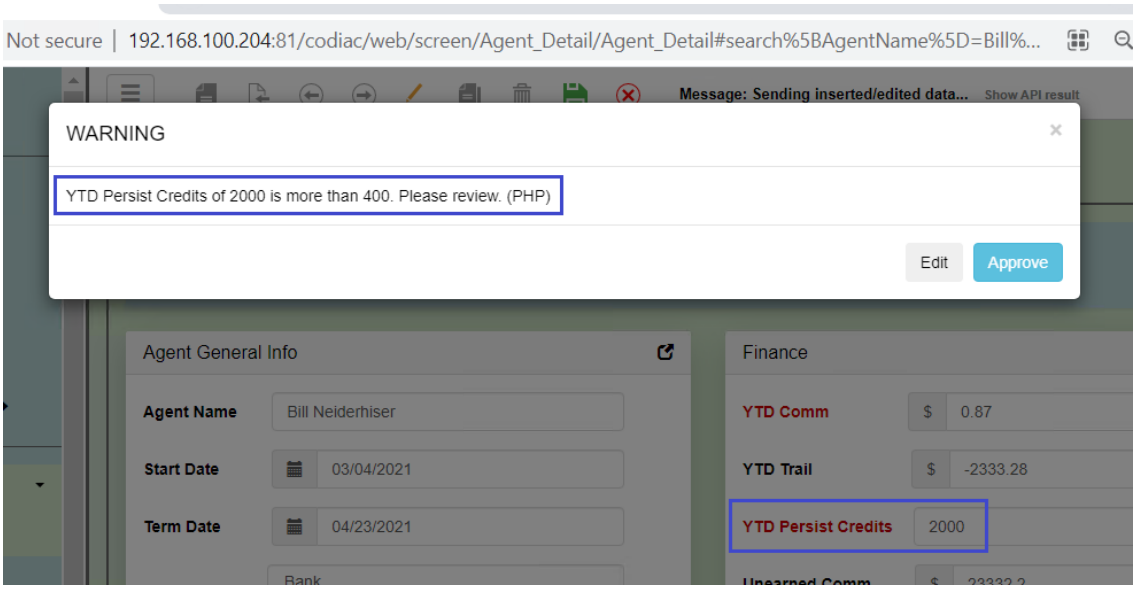




c) Execute screen in Codiac render to test your plugin functions



After entering a number in Persist Credit, then clicking the save icon, the plugin checks if the Persist Credit is over 400. If yes, show message.



4 C++ Alias Framework extension module

The Alias Framework (AF) service can evaluates custom generated aliases or multi generated aliases. This done by going to thru the alias name structure:

AliasType.ModuleName.FunctionName

For example, the "Custom.ExampleModule.CalculateAverage" tells us that the:

- alias type is custom generated alias,
- the extension module file name is `ExampleModule.dll` for MS Windows or `ExampleModule.so` for linux,
- and extension function name is `CalculateAverage`.

Then, the AF loads the extension module from *<service directory>*`\Modules` directory. Finally, executes the extension function which can performs tasks or calculations.

4.1 C++ Alias Framework extension module development

For C++ Alias Framework (AF) extension module, we have decided to use VS Code. It was considered that the user may not have Visual Studio. And AF extension module project can be build with just the AF library and Codiac SDK API library; and without Codiac service source codes.

4.1.1 Required tools

- Microsoft Build Tools for Visual C++ 2019 for Windows
  - Search and download from https://visualstudio.microsoft.com/downloads/
- Cmake 3.12 - https://cmake.org
- Visual Studio Code - https://code.visualstudio.com
- Visual Studio Code: Microsoft C/C++ extension

4.1.2 Visual Studio Code (setup, create module and build module)

Each module has VS Code json files,

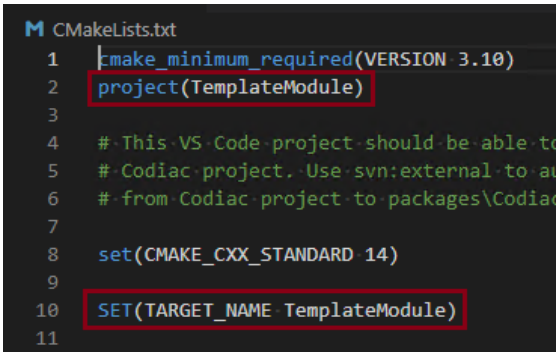
- `.vscode\c_cpp_properties.json` is for intellisense
- `.vscode\tasks.json` is for run tasks and build tasks
- `.vscode\launch.json` is for debugging a running process

So, Visual Studio Code at top level directory can build all modules while at module directory means it can only build on that specific module.

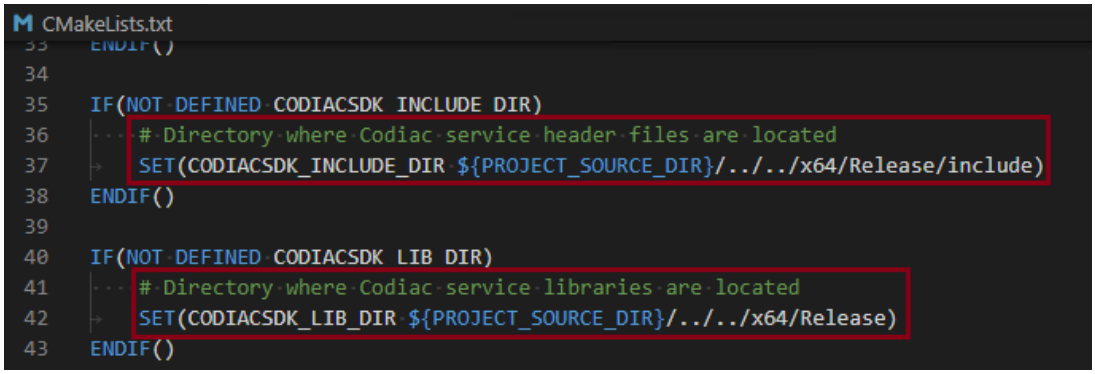
Create a new module

1. Copy the `TemplateModule` directory to new module directory with your module name

2. Change the `TARGET_NAME` and project name of `CMakeLists.txt` to your module name



3. Set the `CODIACSDK_INCLUDE_DIR` to where Codiac service header files are located; and set `CODIACSDK_LIB_DIR` to where Codiac service libraries are located



4. Launch Visual Studio Code
5. Go to **File menu** and **Open Folder** or Ctrl-K Ctrl-O, then open your module directory
6. Add/Edit module functions on `function.cpp`. You can replace or create a new cpp file.

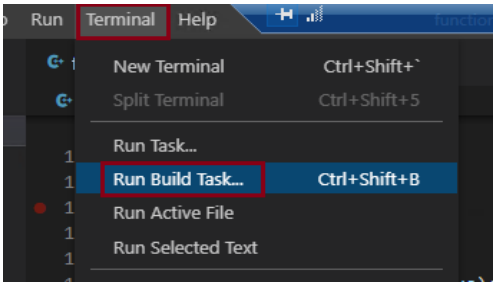
7. Declare it as an C extern function on `function.h` or whatever header file you have created.

```
functions.h > ...
1   /*************************************************
2   /*    @2021 Champion Computer Consulting Inc.  --A
3   /*************************************************
4
5   #pragma once
6
7   #include "Utilities/codiac_module_exports.h"
8
9   extern "C" CODIAC_MODULE_API void MyFunction();
10
```

8. For new cpp files, add the cpp files in `SET(${TARGET_NAME}_SRC` of **`CMakeLists.txt.`**

```
CMakeLists.txt
41
42  SET(${TARGET_NAME}_SRC
43      stdafx.cpp
44      ../Utilities/utility.cpp
45      functions.cpp
46  )
47
```

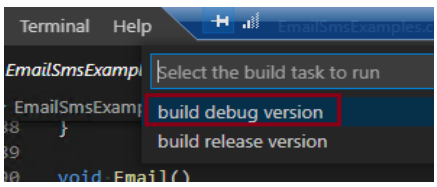
9. Go to **Terminal menu** and **Run Build Task** or Ctrl-Shift-B then select **build release version** task or **build debug version** task to build your module directory



10. If successful, the output module library (dll or so) is in `build/Modules` directory(see build directory section)
11. Copy the output file to *`<service directory>`*`\Modules` directory so that Alias Framework will load and execute this module file when it evaluates custom aliases.

4.1.3 Debugging with Visual Studio Code

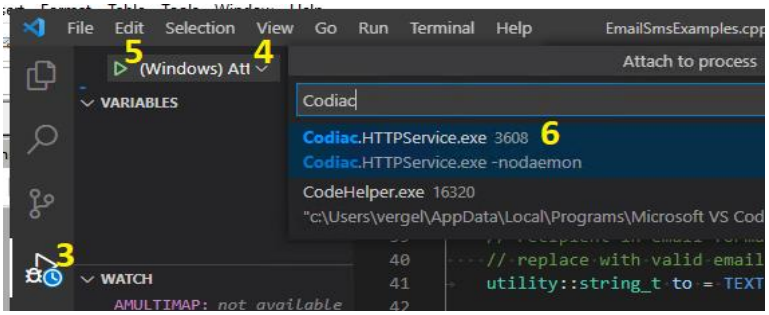
In VS Code, Ctrl-Shift-B then select “build debug version”. Copy the debug build output files to *`<service directory>`*`\Modules` directory.



Then, the run the service build.

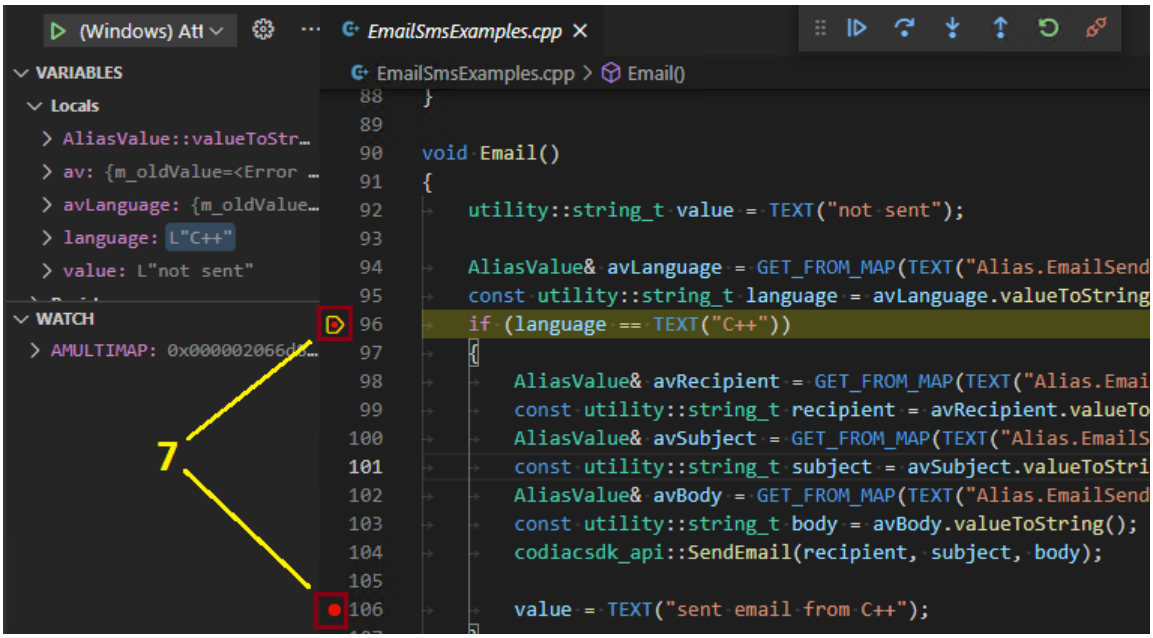
While the service is running, starts VS Code debugger with AF extension module:

1. Run the VS Code in administrator account
2. Open the top level directory or module directory
3. Click the debug icon on left side bar
4. Select the Windows (Attach)
5. Click the green play button (or F5) to start debugging
6. Select Codiac.Service process



At this point, VS Code is attached to the service process and bottom part turns orange.

7. Set a breakpoints on AF extension function
8. Execute a client request that evaluate the custom alias which executes the AF extension function
9. VS Code should stop at break-point and highlight the current line.
10. At this point, you can step through the codes by pressing the F10 (step next), F11 (step into), and F5 (continue) keys or disconnect (orange disconnect icon). You can add watches or see local watches.



4.1.4 Other visual Studio Code Tasks

To activate tasks, go to Terminal menu and select "Run Task...".

- build debug version - on Windows, this produces dll files in build/Modules/Debug directory; on Linux, this produces "so" libraries in `build/Modules` directory.
- build release version - on Windows, this produces dll files in build/Modules/Release directory; on Linux, this produces "so" libraries in `build/Modules` directory.
- clean build - removes build directory

4.1.5 Build directory (out-of-source build)

The VS Code runs build task as an out-of-source build which means that the build directory contains

cmake generated files and build output files. This makes it easier to clean the project build by

just removing the build directory.

Another way to build the project is to manually run cmake to prepares project files and build the project.

For Windows

```
mkdir build
cd build
cmake -G "Visual Studio 16 2019" -T v142,host=x64 -DCMAKE_SYSTEM_VERSION=8.1 ..
cmake --build . --config Debug
```

For Linux

```
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Debug -G "Unix Makefiles" ..
cmake --build .
```

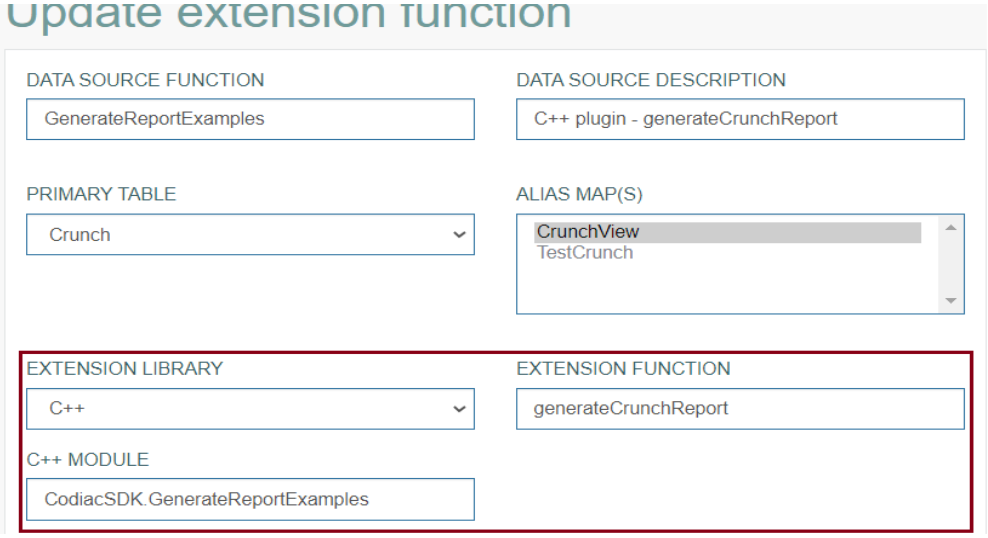
To make release version, change the 'Debug' to 'Release'. For Linux, remove the `-DCMAKE_BUILD_TYPE=Debug`.

One of the files cmake generates is the Visual Studio project file (vcsproj) which can be imported to Visual Studio.

The compiled libraries (dll or so) are in build\Modules directory. On Windows, these libraries are under one more directory. Debug directory for debug version or Release directory for release version.

5 C++ extension functions (plugins)

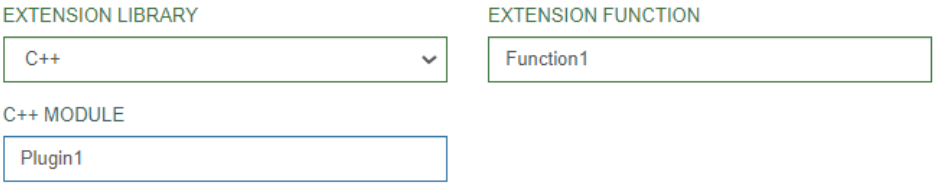
The extension functions are configured in PHP client. Below is an example setup which specifies the module to load and the function to execute.



Let say, we have Plugin1.dll having PluginOne class with Foo method, To set it up as an extension function,

1. In PHP client, go to **System data → Extension functions → Create**.
2. Select **C++** in **EXTENSION LIBRARY**
3. Put filename of module without the extension (Example: “Plugin1”) in **C++ MODULE**
4. Put function name in **EXTENSION FUNCTION** (Example: “Foo”)

The setup will look like below.



Note: Class name (“PluginOne”) is not use in the setup.

5.1 Structure of C++ plugins

The C++ plugins structure follows the Codiac service structure. Meaning that it has `configure` function, `sdkModuleInfo` function, `extController` function, and `extController` function. For convenience, these functions are defined in `BasePlugin` class so that the author of C++ plugin doesn't need to define these functions. So, C++ plugins just need to derived from `BasePlugin` class.

- The `configure` function is executed at start of service. So, any initialization is executed in this function.

- The `sdkModuleInfo` function is also executed at start of service. This function should return the list of plugin functions.
- The `extController` function is called by **`CodiacSDK.Universal`** service when invoke by PHP client. This function calls the plugin function (see 5.1. The plugin class).
- The `funcController` function is not use in plugins.

5.1.1 The plugin class

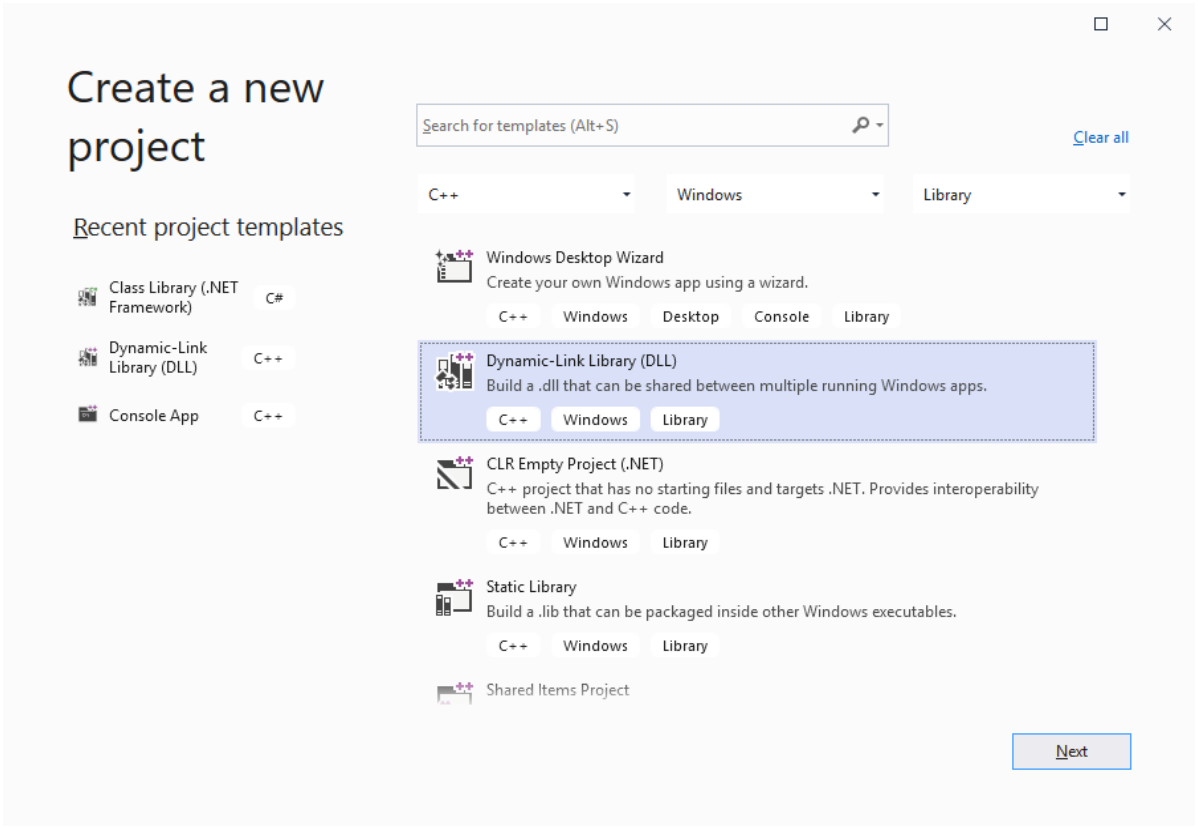
The C++ plugin class derives from BasePlugin class and contains the plugin function implementation. Furthermore, the constructor defines the list of extension functions with bind methods.

For example (image below), the constructor adds “getCrunchMaturityDate” and “getCrunchfData” (with bind methods) to the pluginFunctions map. This map is defined in BasePlugin class. So, when extController function is called with the extension function name (for example: “generateCrunchReport”), the extController function looks for the extension function name in pluginFunction map and calls the bound method.

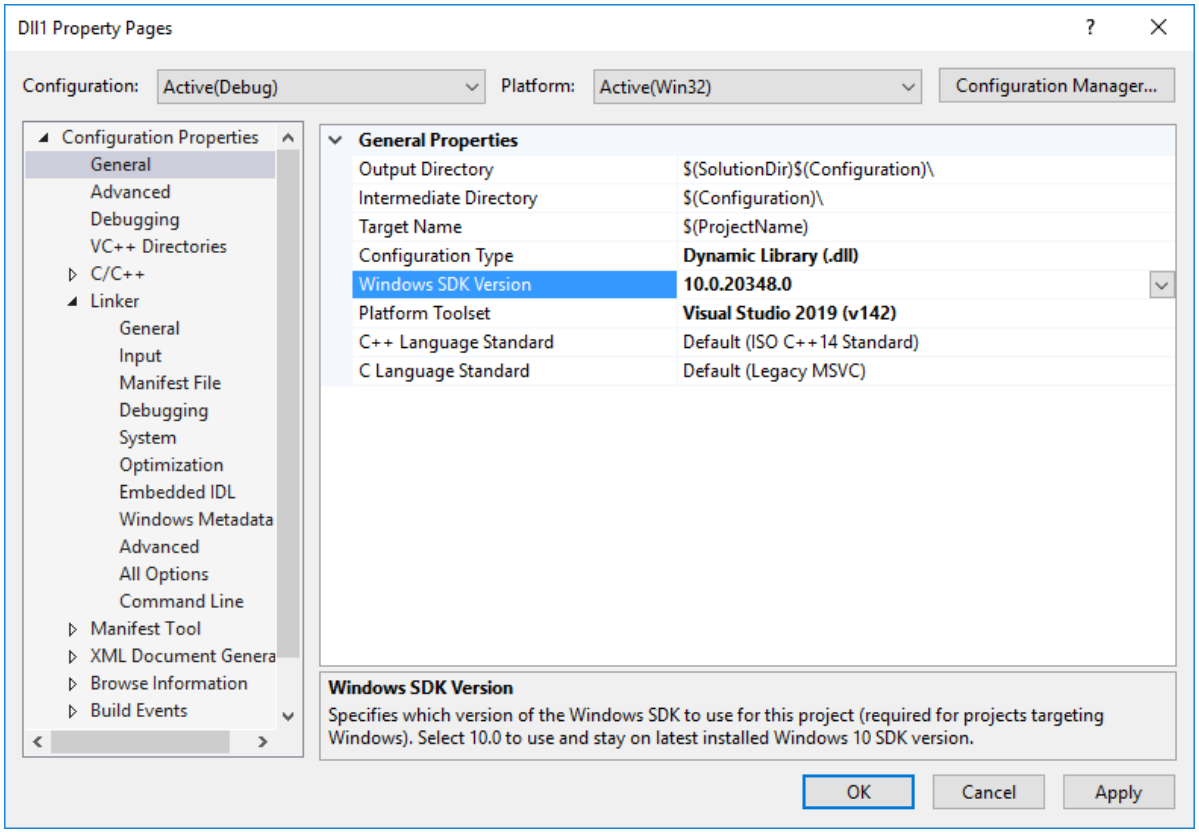
```
AliasFrameworkExamples::AliasFrameworkExamples()
    : BasePlugin("AliasFrameworkExamples")
{
    this->pluginFunctions["getCrunchMaturityDate"] = {
        std::bind(&AliasFrameworkExamples::getCrunchMaturityDate, this, std::placeholders::_1),
        {
            { "func_descr", "plugin AF example that evaluates single value aliases" }
        }
    };
    this->pluginFunctions["getCrunchfData"] = {
        std::bind(&AliasFrameworkExamples::getCrunchfData, this, std::placeholders::_1),
        {
            { "func_descr", "example AF plugin that evaluates multi values aliases" }
        }
    };
```

5.2 How to create a C++ plugin?

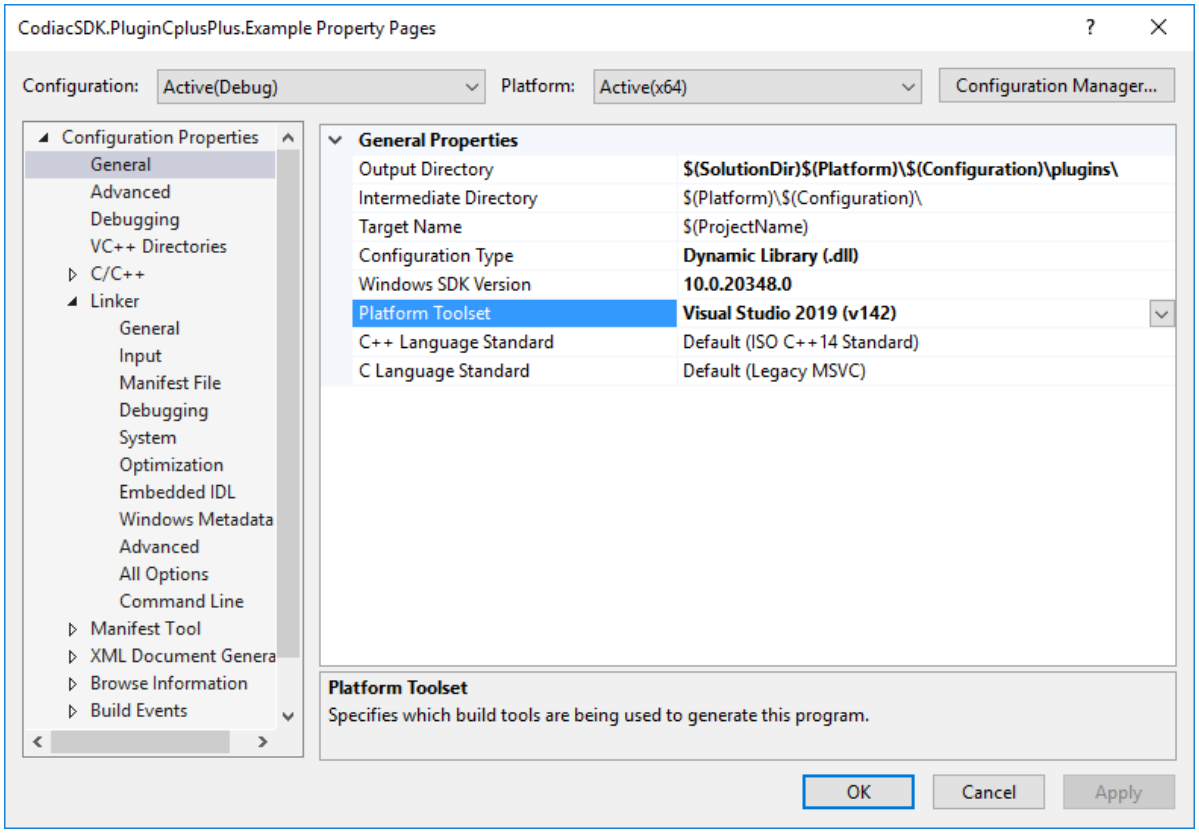
1. Open CodiacSDK.Service solution in Visual Studio (Codiac_v142.sln)
2. Add a project in plugins folder. Project settings:
   a) C++
   b) Windows
   c) Dynamic-Link Library (DLL)



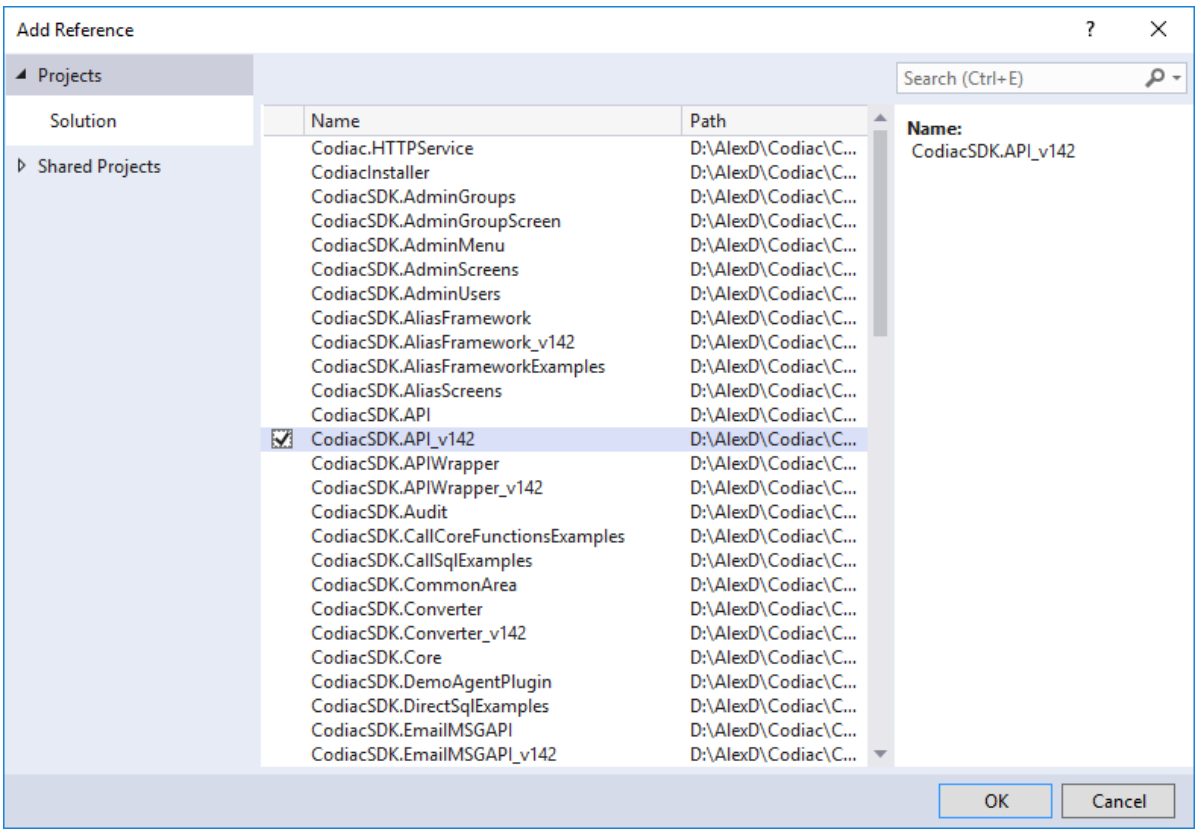
3. Change the project SDK platform version to 10.0.20348.0



4. Change the project platform toolset to v142



5. Update Output directory from “$(SolutionDir)$(Configuration)\” to “$(SolutionDir)$(Platform)\$(Configuration)\plugins\”
6. Add CodiacSDK.API_v142 library to project references as shown below:



7. Add project includes as shown below:

   $(ProjectDir)

   $(SolutionDir)~external\boost\include

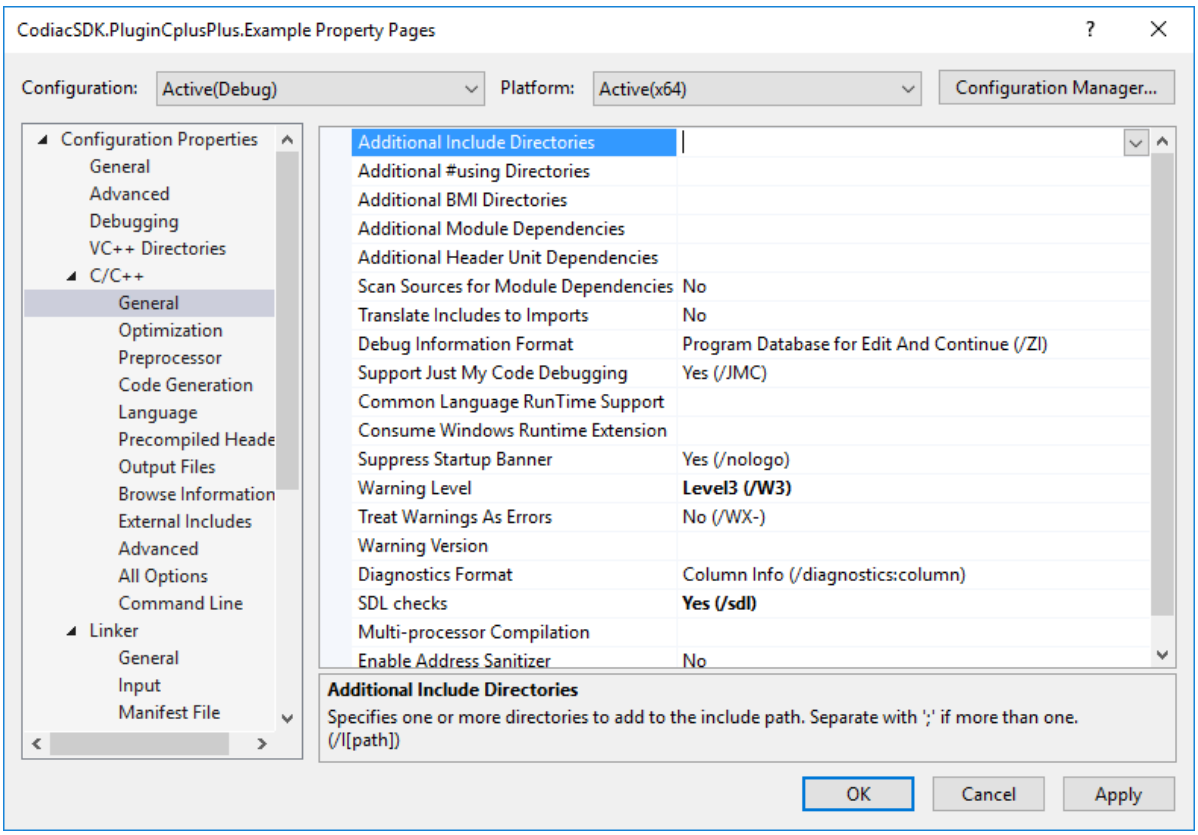
   $(SolutionDir)~external\cpprestsdk\include

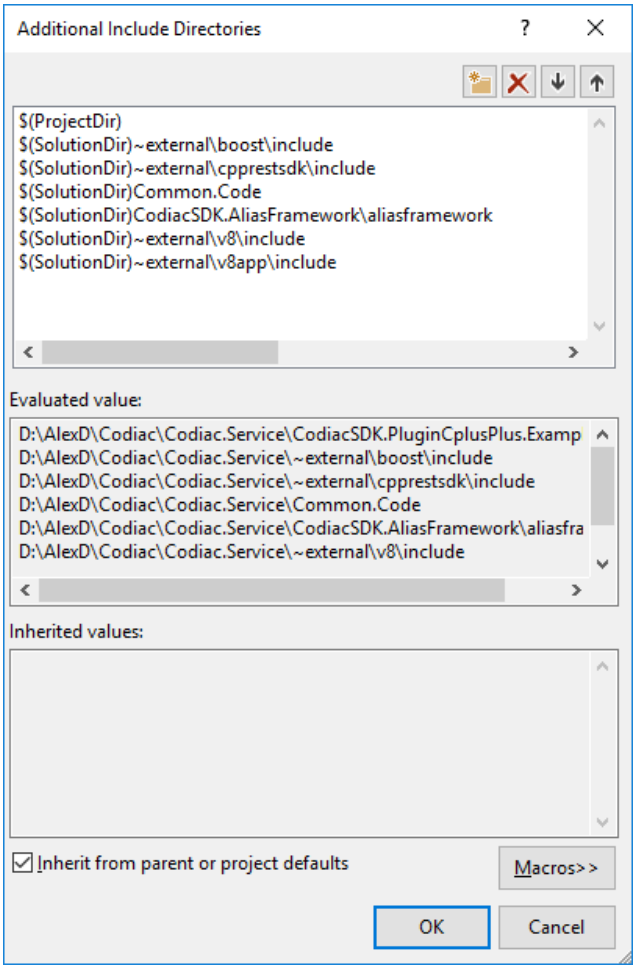
   $(SolutionDir)Common.Code

   $(SolutionDir)CodiacSDK.AliasFramework\aliasframework

   $(SolutionDir)~external\v8\include

   $(SolutionDir)~external\v8app\include

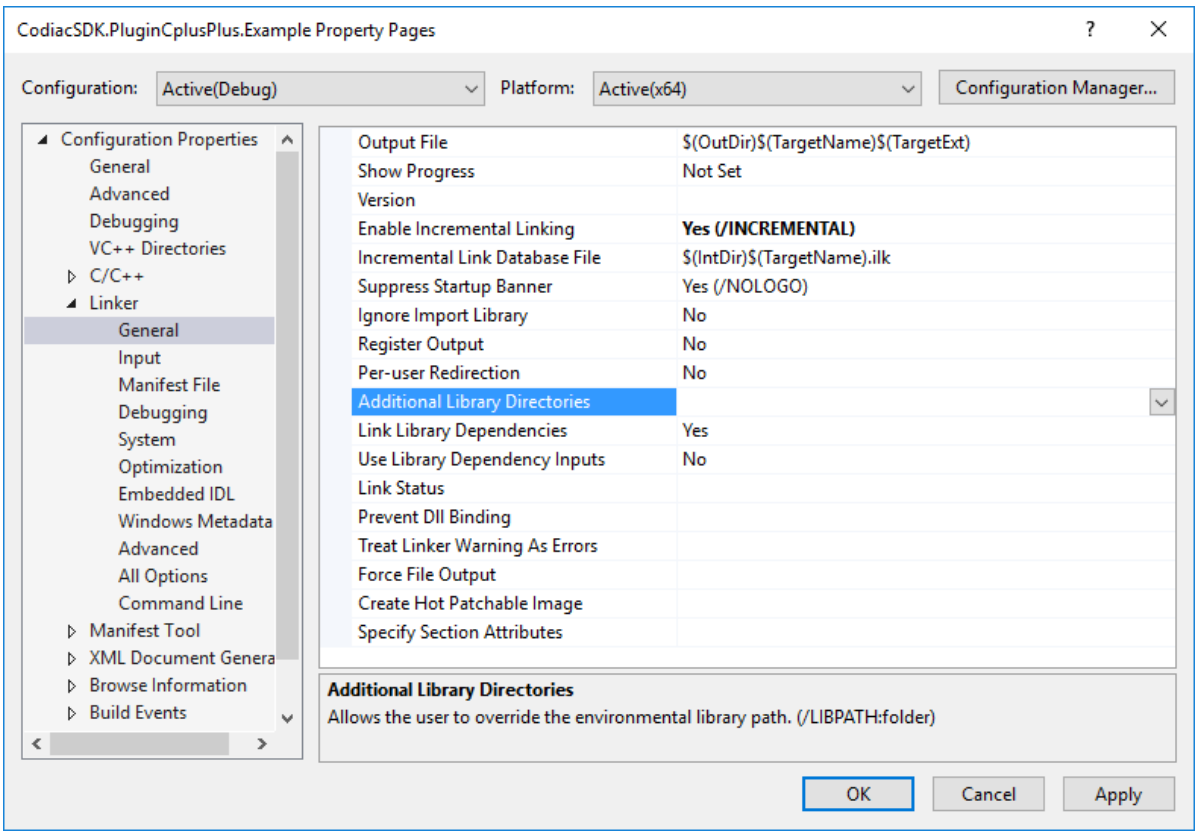

CodiacSDK.PluginCplusPlus.Example Property Pages
Configuration: Active(Debug)
Platform: Active(x64)
Configuration Manager...
Configuration Properties
General
Advanced
Debugging
VC++ Directories
C/C++
General
Optimization
Preprocessor
Code Generation
Language
Precompiled Heade
Output Files
Browse Information
External Includes
Advanced
All Options
Command Line
Linker
General
Input
Manifest File
Additional Include Directories
Additional #using Directories
Additional BMI Directories
Additional Module Dependencies
Additional Header Unit Dependencies
Scan Sources for Module Dependencies No
Translate Includes to Imports No
Debug Information Format Program Database for Edit And Continue (/ZI)
Support Just My Code Debugging Yes (/JMC)
Common Language RunTime Support
Consume Windows Runtime Extension
Suppress Startup Banner Yes (/nologo)
Warning Level Level3 (/W3)
Treat Warnings As Errors No (/WX-)
Warning Version
Diagnostics Format Column Info (/diagnostics:column)
SDL checks Yes (/sdl)
Multi-processor Compilation
Enable Address Sanitizer No
Additional Include Directories
Specifies one or more directories to add to the include path. Separate with ';' if more than one.
(/I[path])
OK
Cancel
Apply

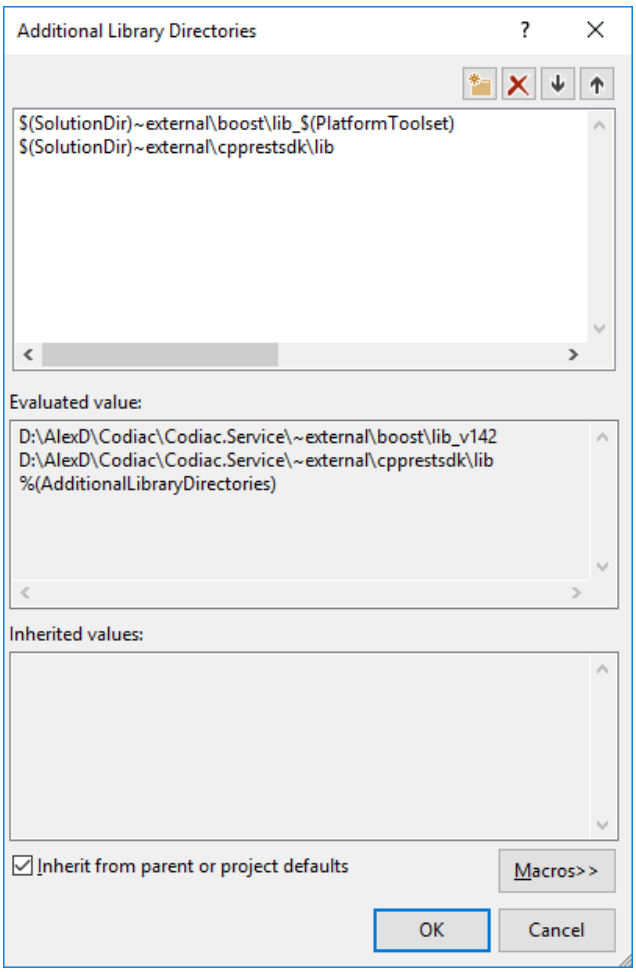


8. Add project library directories:

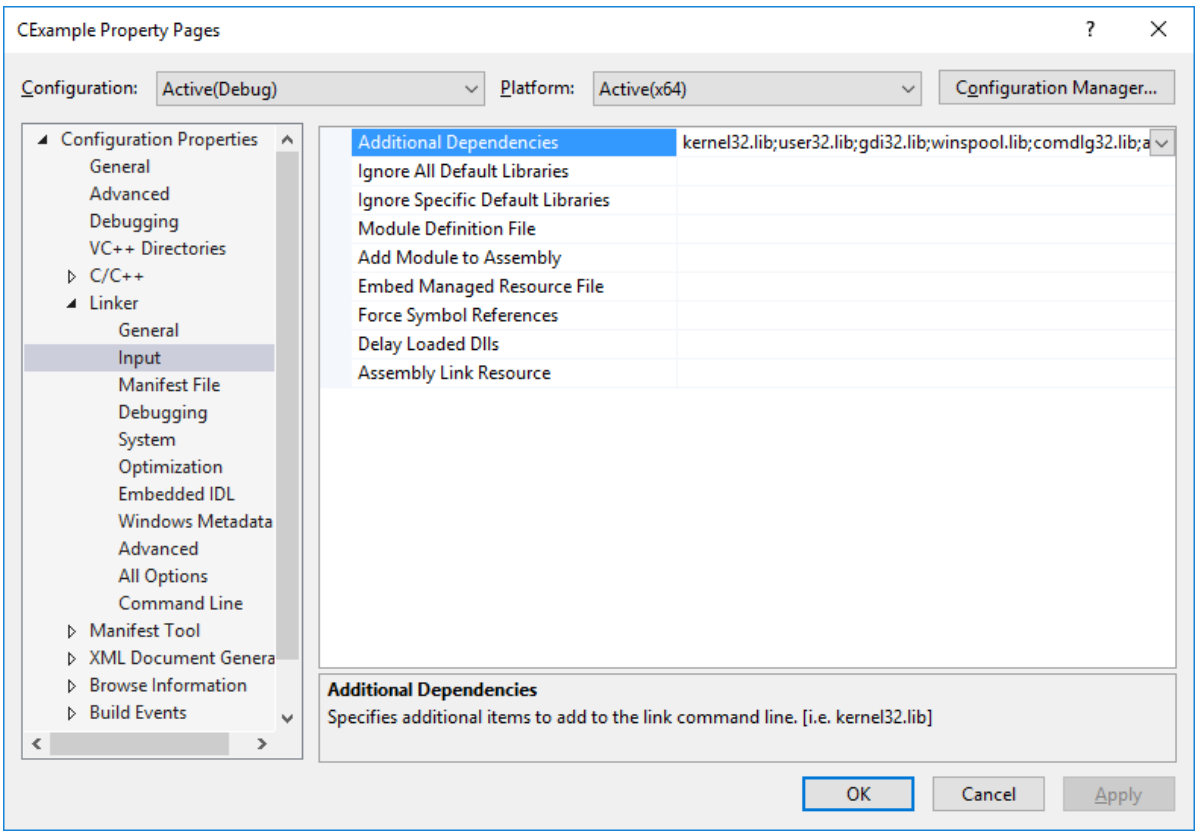
   $(SolutionDir)~external\boost\lib_$(PlatformToolset)

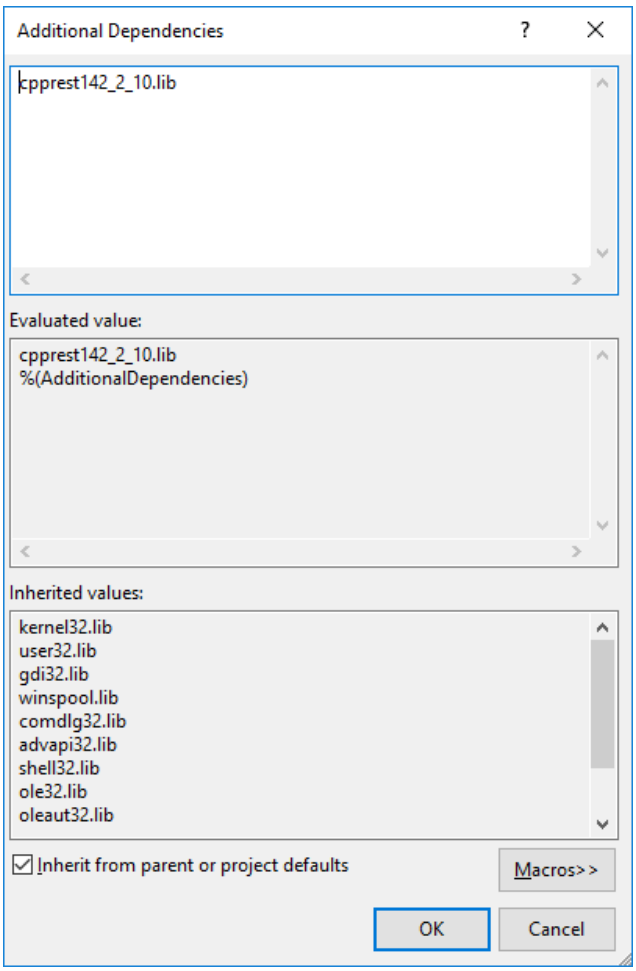
   $(SolutionDir)~external\cpprestsdk\lib


CodiacSDK.PluginCplusPlus.Example Property Pages
Configuration: Active(Debug)
Platform: Active(x64)
Configuration Manager...
Configuration Properties
General
Advanced
Debugging
VC++ Directories
C/C++
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Windows Metadata
Advanced
All Options
Command Line
Manifest Tool
XML Document Genera
Browse Information
Build Events
Output File $(OutDir)$(TargetName)$(TargetExt)
Show Progress Not Set
Version
Enable Incremental Linking Yes (/INCREMENTAL)
Incremental Link Database File $(IntDir)$(TargetName).ilk
Suppress Startup Banner Yes (/NOLOGO)
Ignore Import Library No
Register Output No
Per-user Redirection No
Additional Library Directories
Link Library Dependencies Yes
Use Library Dependency Inputs No
Link Status
Prevent Dll Binding
Treat Linker Warning As Errors
Force File Output
Create Hot Patchable Image
Specify Section Attributes
Additional Library Directories
Allows the user to override the environmental library path. (/LIBPATH:folder)
OK
Cancel
Apply



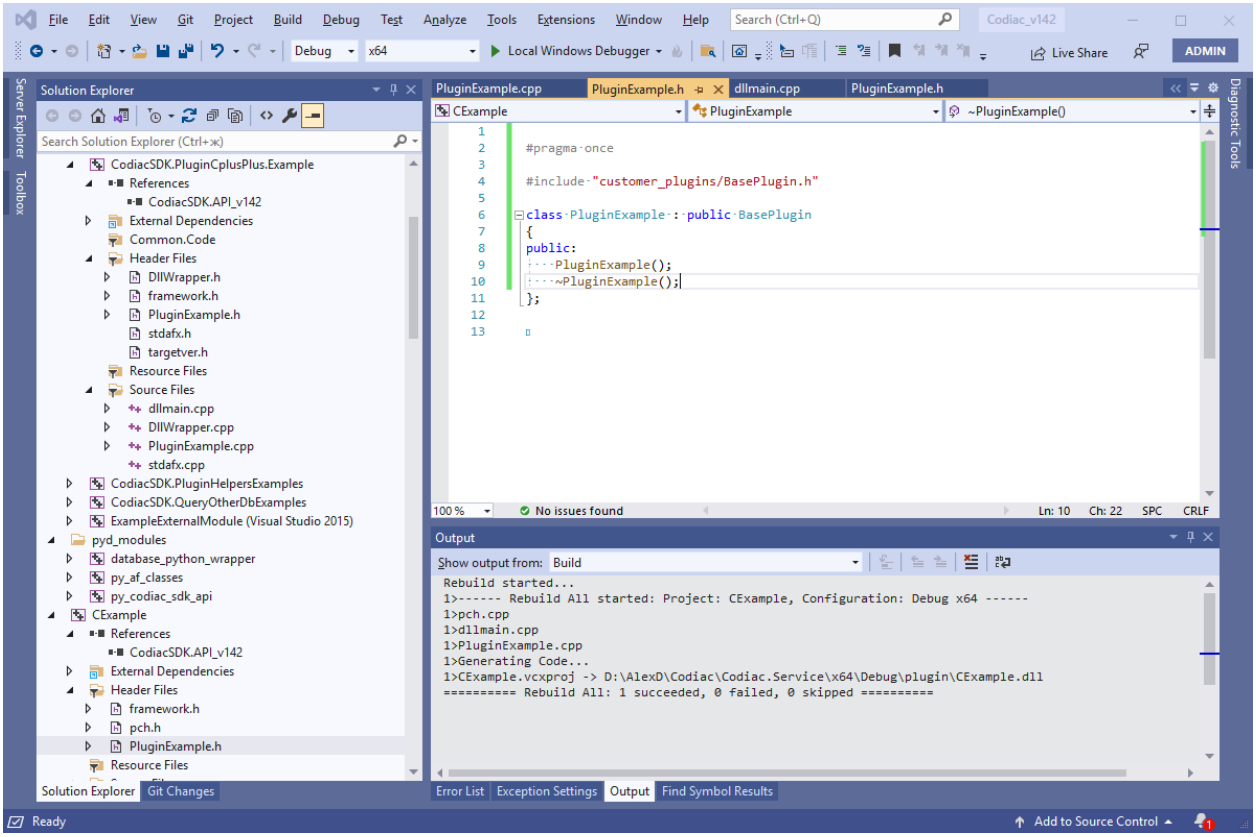
9. Add project dependencies:
   - for debug configuration: `cpprest142_2_10d.lib`
   - for release configuration: `cpprest142_2_10.lib`


CExample Property Pages
Configuration: Active(Debug)
Platform: Active(x64)
Configuration Manager...
Configuration Properties
General
Advanced
Debugging
VC++ Directories
C/C++
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Windows Metadata
Advanced
All Options
Command Line
Manifest Tool
XML Document Genera
Browse Information
Build Events
Additional Dependencies
kernel32.lib;user32.lib;gdi32.lib;winspool.lib;comdlg32.lib;a
Ignore All Default Libraries
Ignore Specific Default Libraries
Module Definition File
Add Module to Assembly
Embed Managed Resource File
Force Symbol References
Delay Loaded Dlls
Assembly Link Resource
Additional Dependencies
Specifies additional items to add to the link command line. [i.e. kernel32.lib]
OK
Cancel
Apply

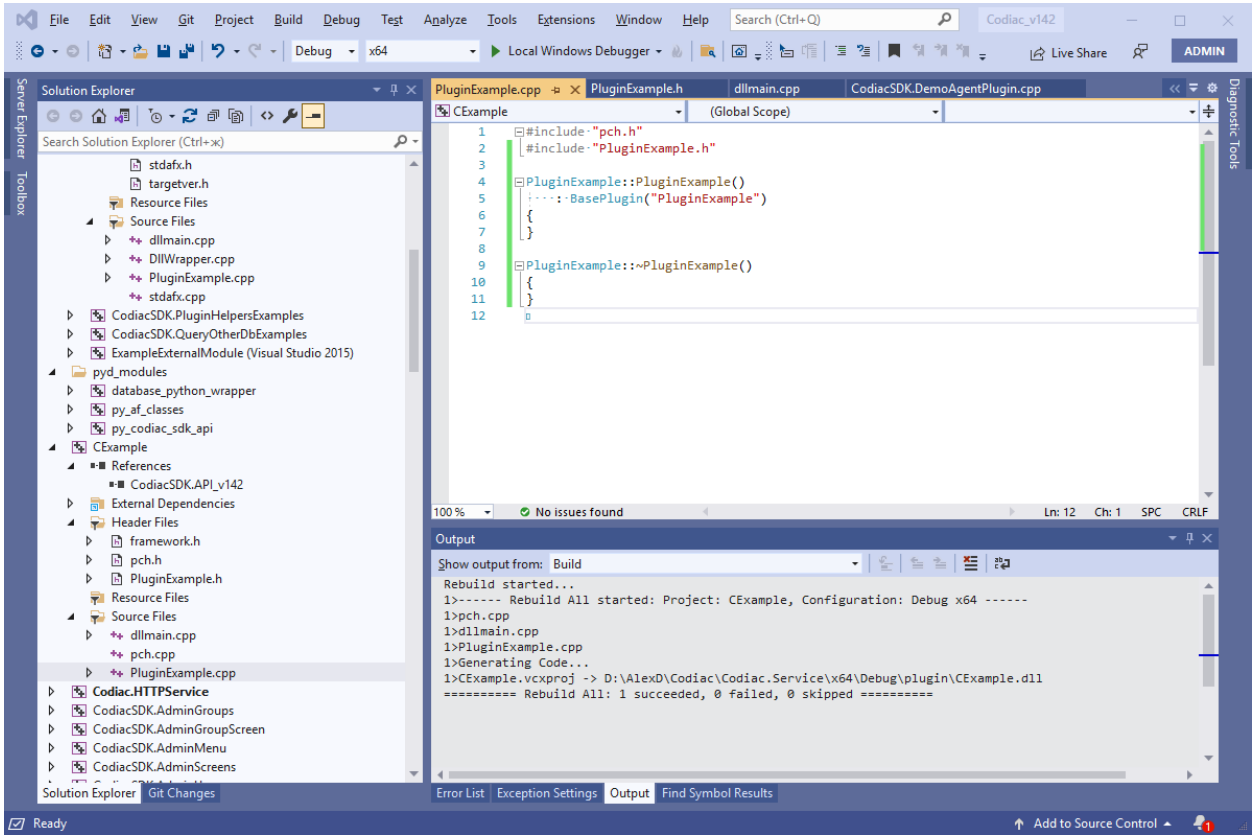


10. Create a class derived from BasePlugin class and create instance with plugin name. Below is an example class (PluginExample):

```
#pragma once

#include "customer_plugins/BasePlugin.h"

class PluginExample : public BasePlugin
{
public:
    PluginExample();
    ~PluginExample();
};
```

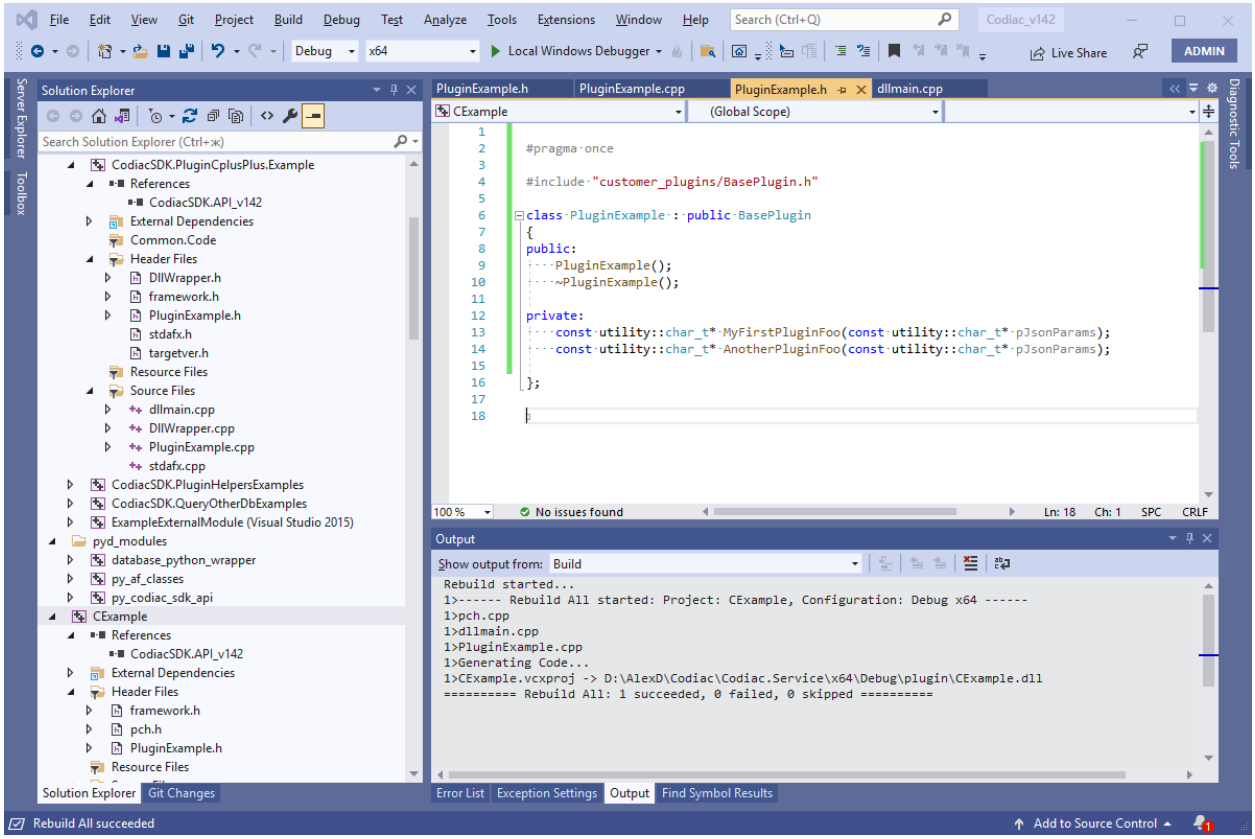



```
#include "pch.h"
#include "PluginExample.h"

PluginExample::PluginExample()
    : BasePlugin("PluginExample")
{
}

PluginExample::~PluginExample()
{
}
```

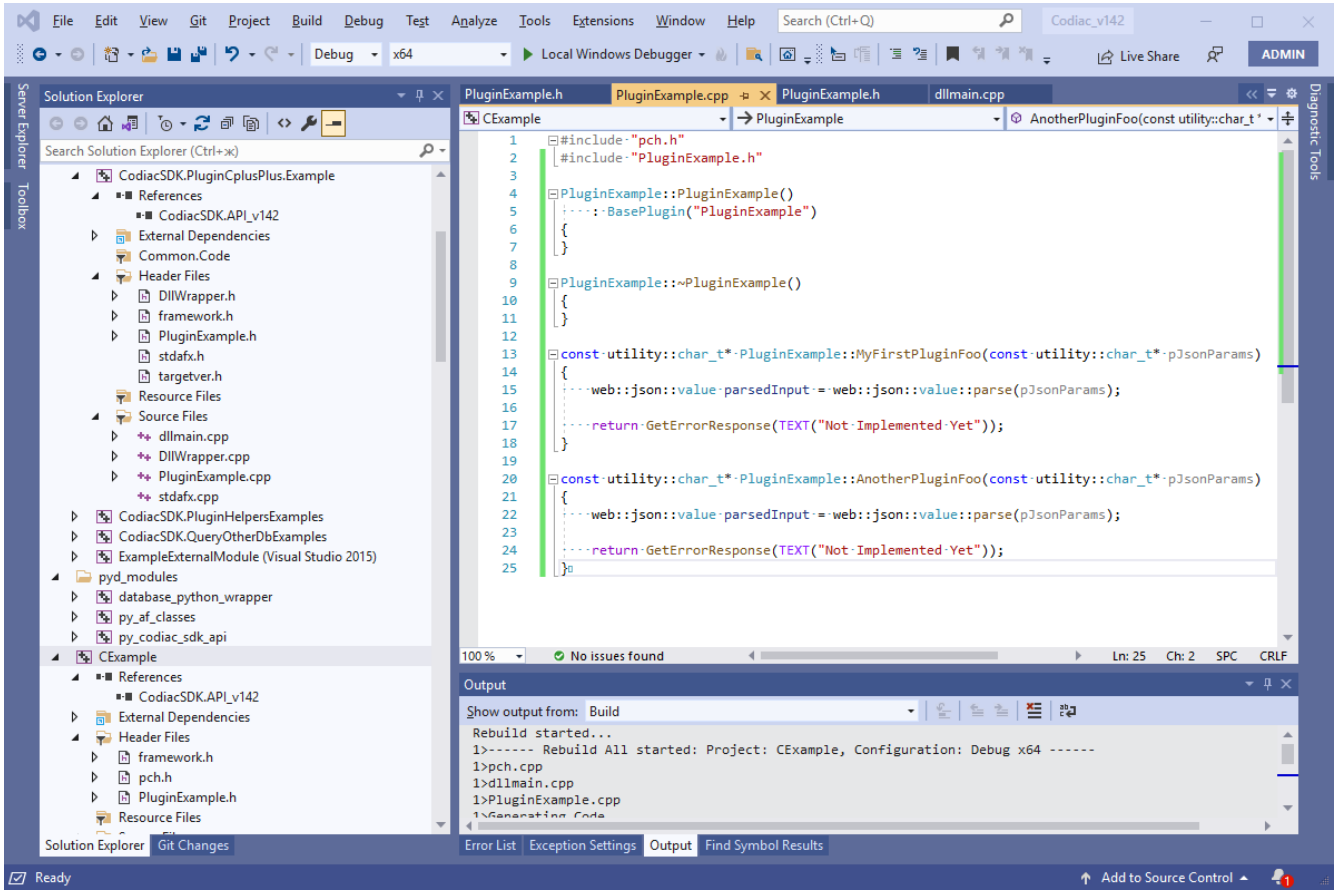


11. Create methods for the plugin class

```
private:
    const utility::char_t* MyFirstPluginFoo(const utility::char_t* pJsonParams);
    const utility::char_t* AnotherPluginFoo(const utility::char_t* pJsonParams);
```

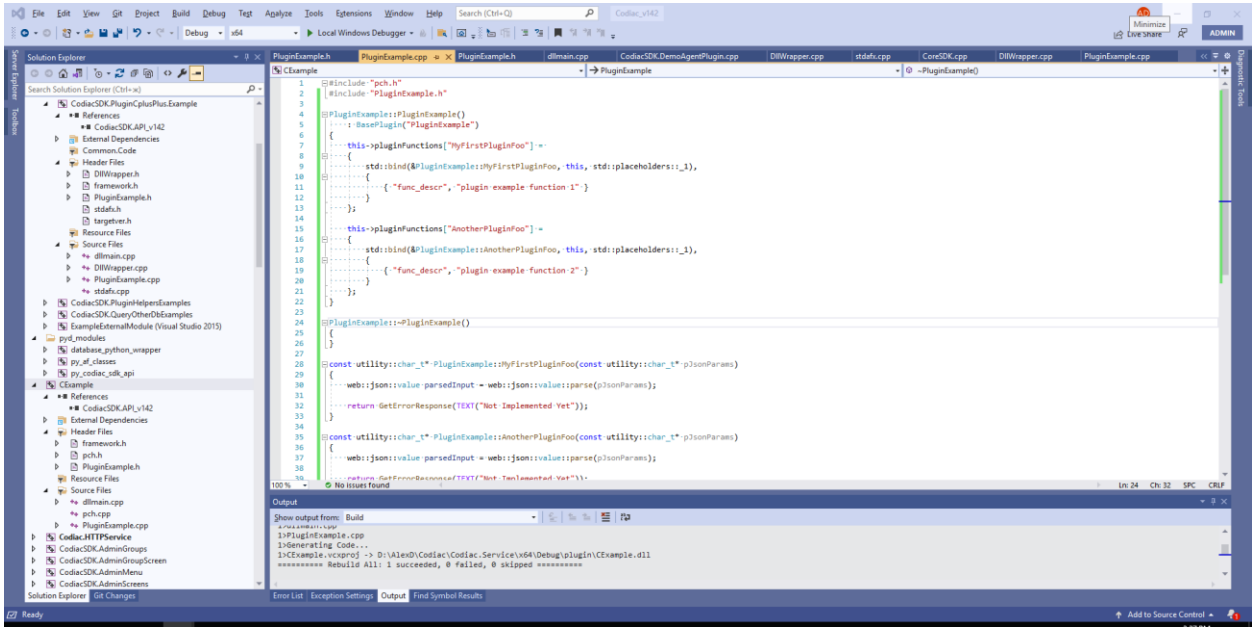



```
const utility::char_t* PluginExample::MyFirstPluginFoo(
       const utility::char_t* pJsonParams)
{
    web::json::value parsedInput = web::json::value::parse(pJsonParams);

    return GetErrorResponse(TEXT("Not Implemented Yet"));
}

const utility::char_t* PluginExample::AnotherPluginFoo(
       const utility::char_t* pJsonParams)
{
    web::json::value parsedInput = web::json::value::parse(pJsonParams);

    return GetErrorResponse(TEXT("Not Implemented Yet"));
}
```



12. Add the new methods to pluginFunctions map. This means to add the plugin function name and 19 the bind the method. Don't declare the pluginFunction map in plugin class. This map is already declared in BasePlugin. Follow the structure below

```
this->pluginFunctions["MyFirstPluginFoo"] =
{
    std::bind(&PluginExample::MyFirstPluginFoo, this, std::placeholders::_1),
    {
        { "func_descr", "plugin example function 1" }
    }
};

this->pluginFunctions["AnotherPluginFoo"] =
{
    std::bind(&PluginExample::AnotherPluginFoo, this, std::placeholders::_1),
    {
        { "func_descr", "plugin example function 2" }
    }
};
```



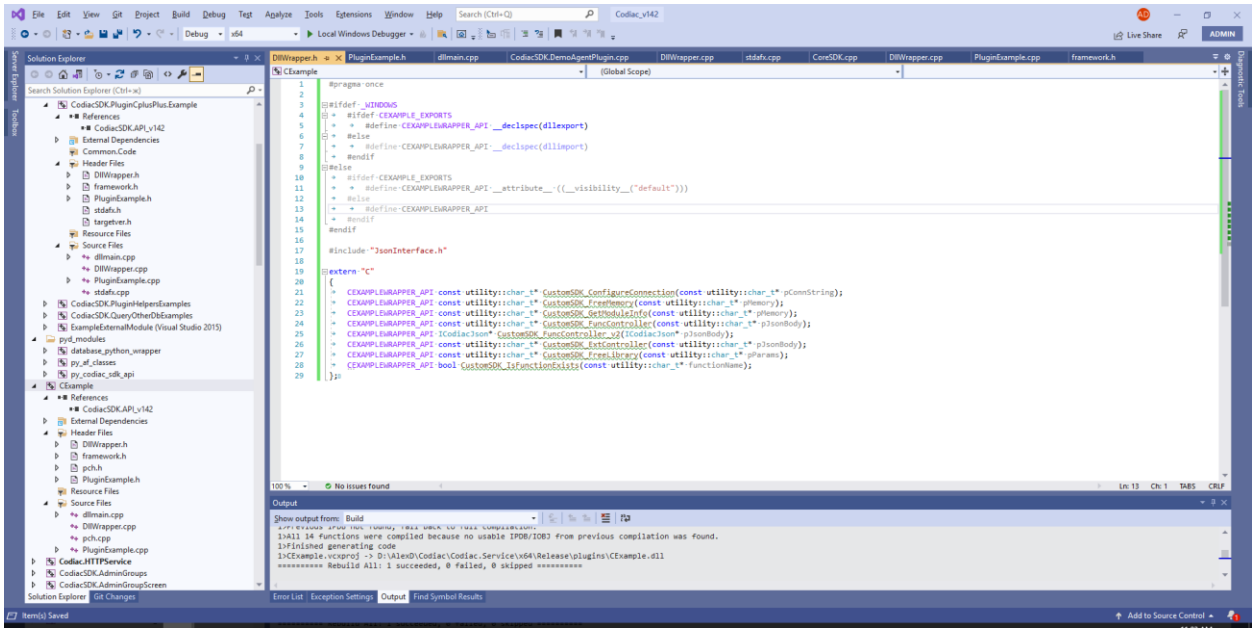
13. Declare DLL external plugin interfaces

a) Create a class `DllWrapper` and replace it content to following code:

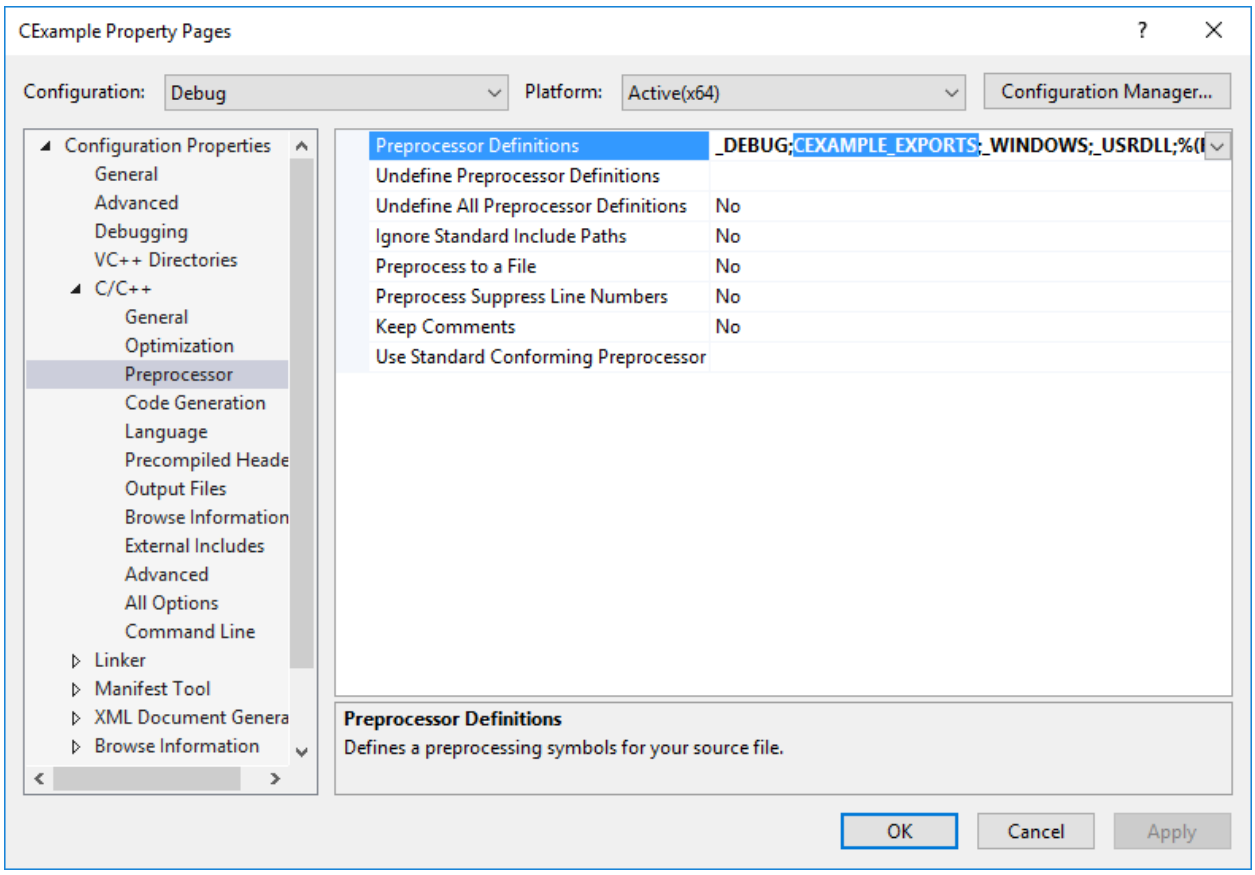
- `DllWrapper.h`

```
#pragma once

#ifdef _WINDOWS
	#ifdef CEXAMPLE_EXPORTS
		#define CEXAMPLEWRAPPER_API __declspec(dllexport)
	#else
		#define CEXAMPLEWRAPPER_API __declspec(dllimport)
	#endif
#else
	#ifdef CEXAMPLE_EXPORTS
		#define CEXAMPLEWRAPPER_API __attribute__ ((__visibility__("default")))
	#else
		#define CEXAMPLEWRAPPER_API
	#endif
#endif

#include "JsonInterface.h"

extern "C"
{
	CEXAMPLEWRAPPER_API const utility::char_t * CustomSDK_ConfigureConnection(
		const utility::char_t * pConnString);
	CEXAMPLEWRAPPER_API const utility::char_t * CustomSDK_FreeMemory(
		const utility::char_t * pMemory);
	CEXAMPLEWRAPPER_API const utility::char_t * CustomSDK_GetModuleInfo(
		const utility::char_t * pMemory);
	CEXAMPLEWRAPPER_API const utility::char_t * CustomSDK_FuncController(
		const utility::char_t * pJsonBody);
	CEXAMPLEWRAPPER_API ICodiacJson* CustomSDK_FuncController_v2(
		ICodiacJson* pJsonBody);
	CEXAMPLEWRAPPER_API const utility::char_t * CustomSDK_ExtController(
```

```
        const utility::char_t * pJsonBody);
    CEXAMPLEWRAPPER_API const utility::char_t * CustomSDK_FreeLibrary(
        const utility::char_t * pParams);
    CEXAMPLEWRAPPER_API bool CustomSDK_IsFunctionExists(
        const utility::char_t* functionName);
};
```



Where CEXAMPLE_EXPORTS declared in your project configuration preprocessor definition, so you need rename it to your value. Also CEXAMPLEWRAPPER_API should be renamed too.



Example:

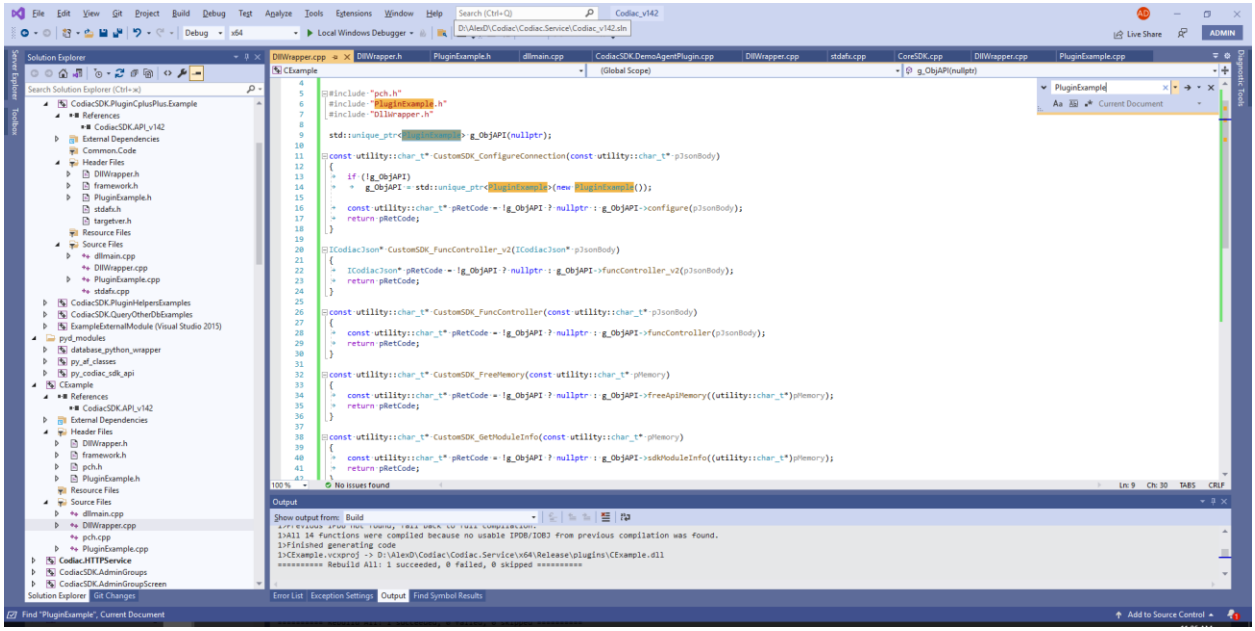
If your plugin name is `CalculatorPlugin`, in preprocessor you will have `CALCULATORPLUGIN_EXPORTS`. In this case you need

- rename CEXAMPLE_EXPORTS to CALCULATORPLUGIN_EXPORTS
- rename CEXAMPLEWRAPPER_API to CALCULATORPLUGINWRAPPER_API

- `DllWrapper.cpp`

```
#include "pch.h"
#include "PluginExample.h"
#include "DllWrapper.h"

std::unique_ptr<PluginExample> g_ObjAPI(nullptr);

const utility::char_t* CustomSDK_ConfigureConnection(const utility::char_t*
pJsonBody)
{
	if (!g_ObjAPI)
		g_ObjAPI = std::unique_ptr<PluginExample>(new PluginExample());

	const utility::char_t* pRetCode = !g_ObjAPI ? nullptr : g_ObjAPI-
>configure(pJsonBody);
	return pRetCode;
}

ICodiacJson* CustomSDK_FuncController_v2(ICodiacJson* pJsonBody)
{
	ICodiacJson* pRetCode = !g_ObjAPI ? nullptr : g_ObjAPI-
```

```
>funcController_v2(pJsonBody);
	return pRetCode;
}

const utility::char_t* CustomSDK_FuncController(const utility::char_t* pJsonBody)
{
	const utility::char_t* pRetCode = !g_ObjAPI ? nullptr : g_ObjAPI-
>funcController(pJsonBody);
	return pRetCode;
}

const utility::char_t* CustomSDK_FreeMemory(const utility::char_t* pMemory)
{
	const utility::char_t* pRetCode = !g_ObjAPI ? nullptr : g_ObjAPI-
>freeApiMemory((utility::char_t*)pMemory);
	return pRetCode;
}

const utility::char_t* CustomSDK_GetModuleInfo(const utility::char_t* pMemory)
{
	const utility::char_t* pRetCode = !g_ObjAPI ? nullptr : g_ObjAPI-
>sdkModuleInfo((utility::char_t*)pMemory);
	return pRetCode;
}

const utility::char_t* CustomSDK_ExtController(const utility::char_t* pJsonBody)
{
	const utility::char_t* pRetCode = !g_ObjAPI ? nullptr : g_ObjAPI-
>extController(pJsonBody);
	return pRetCode;
}

const utility::char_t* CustomSDK_FreeLibrary(const utility::char_t* pParams)
{
	const utility::char_t* pRetCode = !g_ObjAPI ? nullptr : g_ObjAPI-
>freeLibrary(pParams);
	return pRetCode;
}

bool CustomSDK_IsFunctionExists(const utility::char_t* functionName)
{
	return !g_ObjAPI ? false : g_ObjAPI-
>isFunctionExists((utility::char_t*)functionName);
}
```



Where **PluginExample** name should be replaced to your plugin class name (in our example the class name is **`PluginExample`**)

14. Build the solution or just the plugin project. Copy the generated plugin module (dll or so) to *`<service directory>`*`\plugins` directory.
15. Finally, execute the plugin with client.

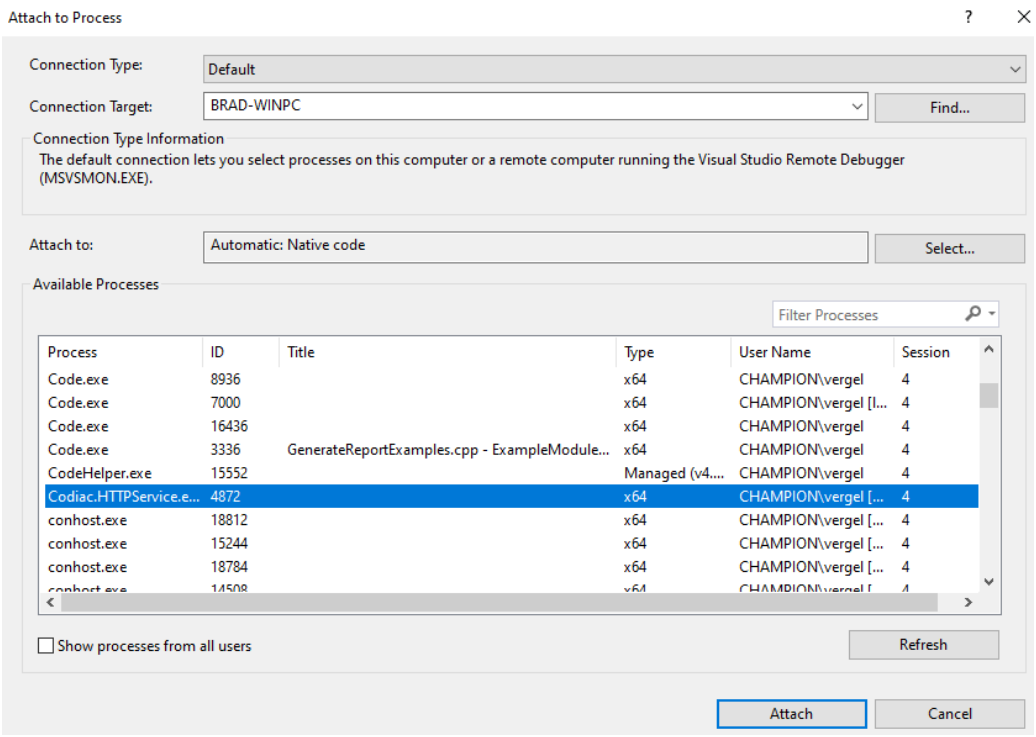
5.3 Debugging C++ plugin

Prerequisite before staring the debugger:

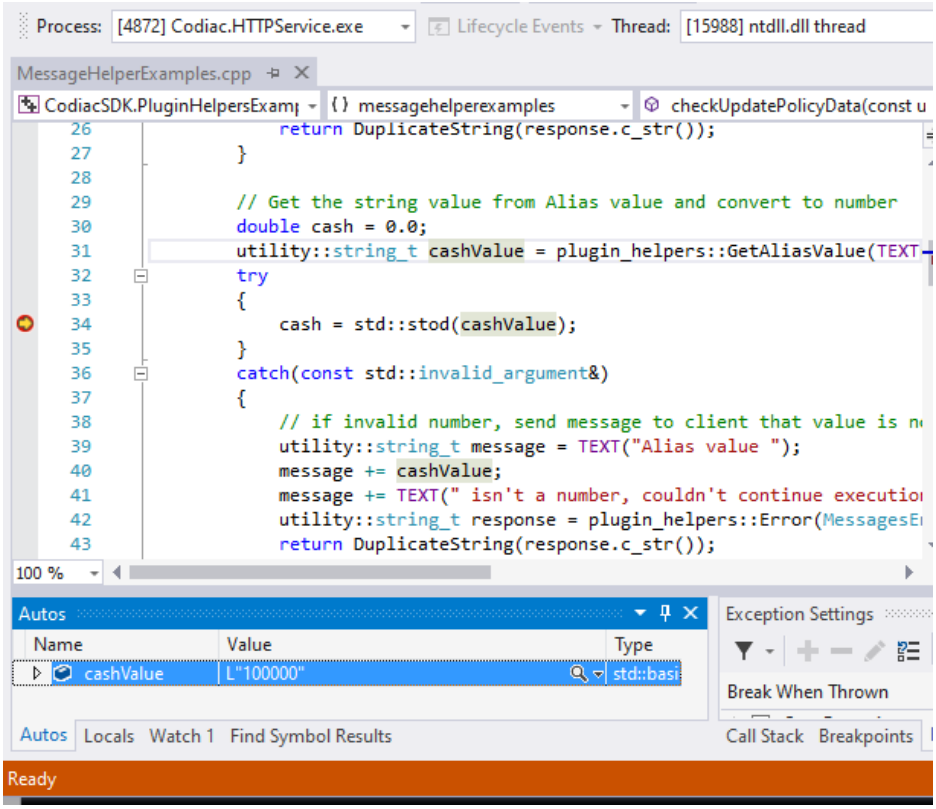
- the plugin module (dll or so) is in plugin directory.
- the plugin source code must match the plugin module
- service is running

Steps to attach debugger to C++ plugin:

1. Attached Visual Studio debugger to Codiac service process (Ctrl-Alt-P).



2. Open plugin class and set breakpoint
3. Execute the extension function in client. This should stop at the breakpoint and current line will have yellow arrow.



4. You should be able to step through the code be pressing the F10 (step over) and F11 (step into) keys and see the variable values in watches.

5. To end debugging, go to **Debug** menu and select **Detach all**.

6 Remote Debugging C++ plugin and AF extension functions

Full instruction for the remote debugging for the C++ applications can be found on the Remote Debug a C++ Project - Visual Studio (Windows) | Microsoft Learn page.

6.1 Required tools

Download and install the tools below on the remote machine:

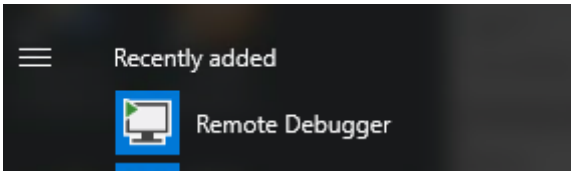
- **Remote Debugger**
  The Remote Debugger can be downloaded at the following link: Remote Debug a C++ Project - Visual Studio (Windows) | Microsoft Learn

6.2 Tool configurations

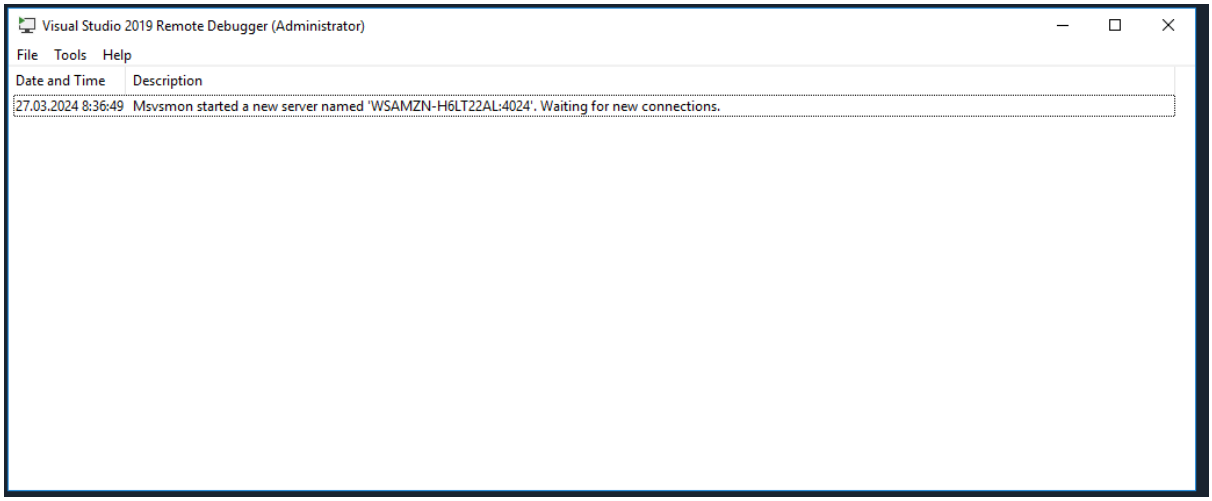
6.2.1 Install Remote Debugger

To set up the tool for remote debugging, follow the steps below:

1. Download the Remote Debugger.
2. Install the Remote Debugger on the remote machine.
3. Launch the Remote Debugger on the remote machine.

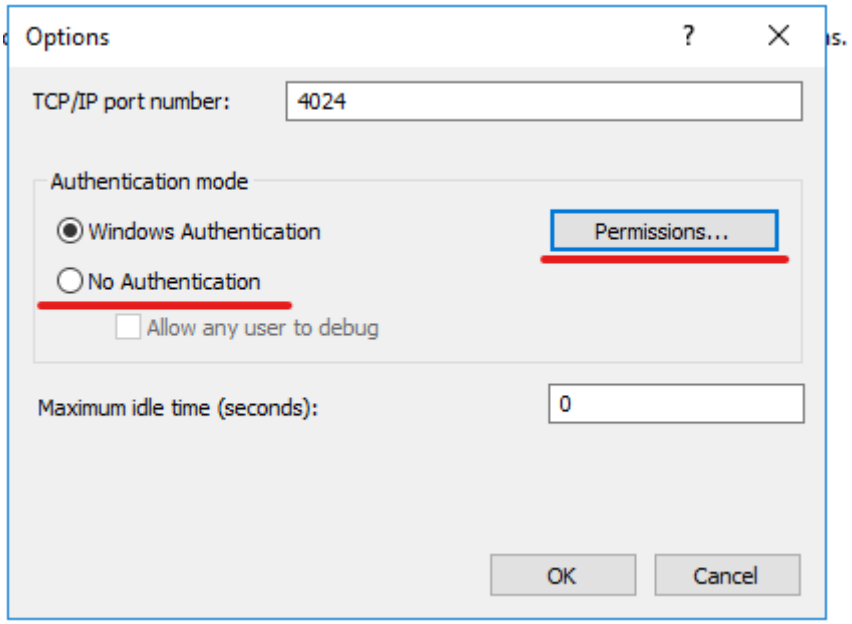


The Remote Debugger working screen will be displayed as follows:



In the Remote Debugger working screen, perform the following actions, to configure the access to the Remote Debugger:

1. In the Menu, click the **Tools** menu section and select the **Options…** menu item. The Options pop-up window opens.



In the Options pop-up window, fill in the necessary fields:

- **TCP/IP port number** – the communication endpoints that applications use to coordinate and establish communication over a network using the TCP/IP protocol suite.

Based on the desired security level select:

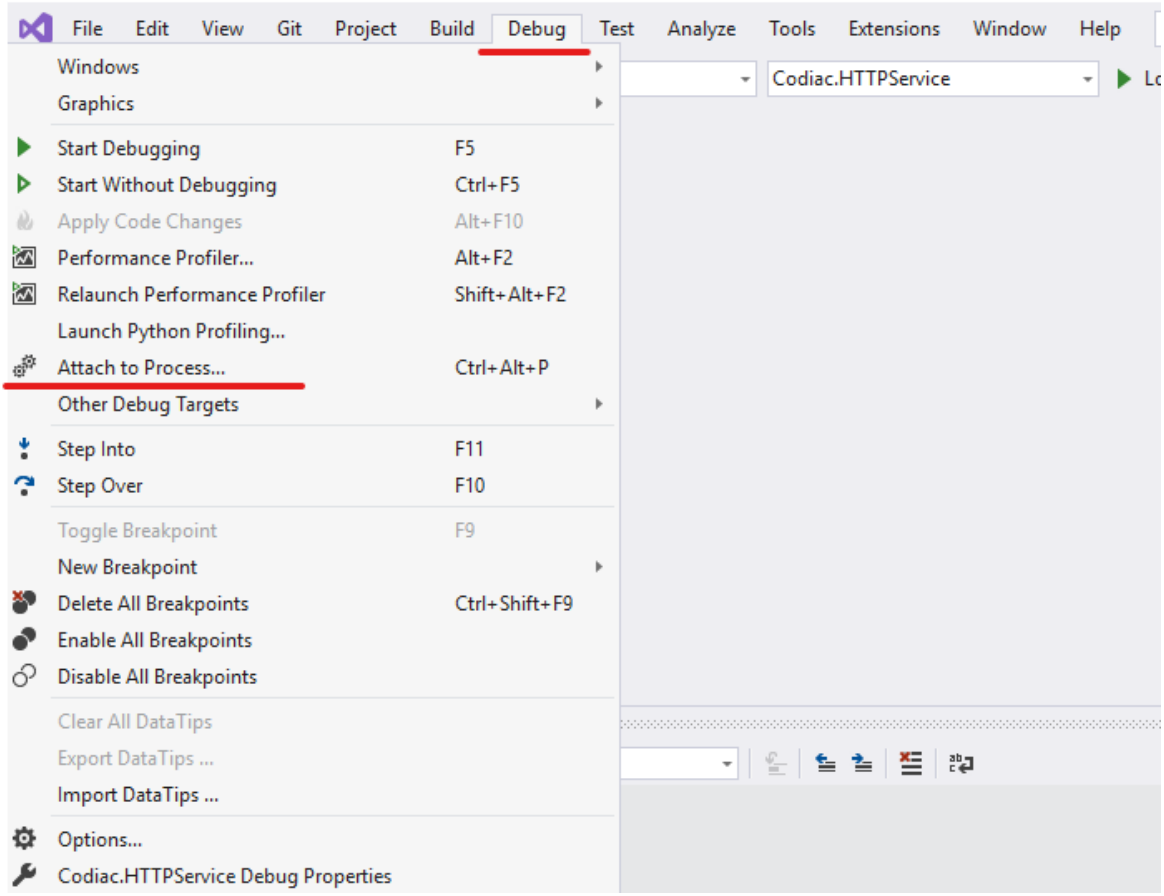
- the **Windows Authentication** access and set the **Permissions** to a user,
  or
- the **No Authentication** access for the remote debugging.

After the above-mentioned options are set, the remote machine will be ready to accept the Remote Debugger's requests.

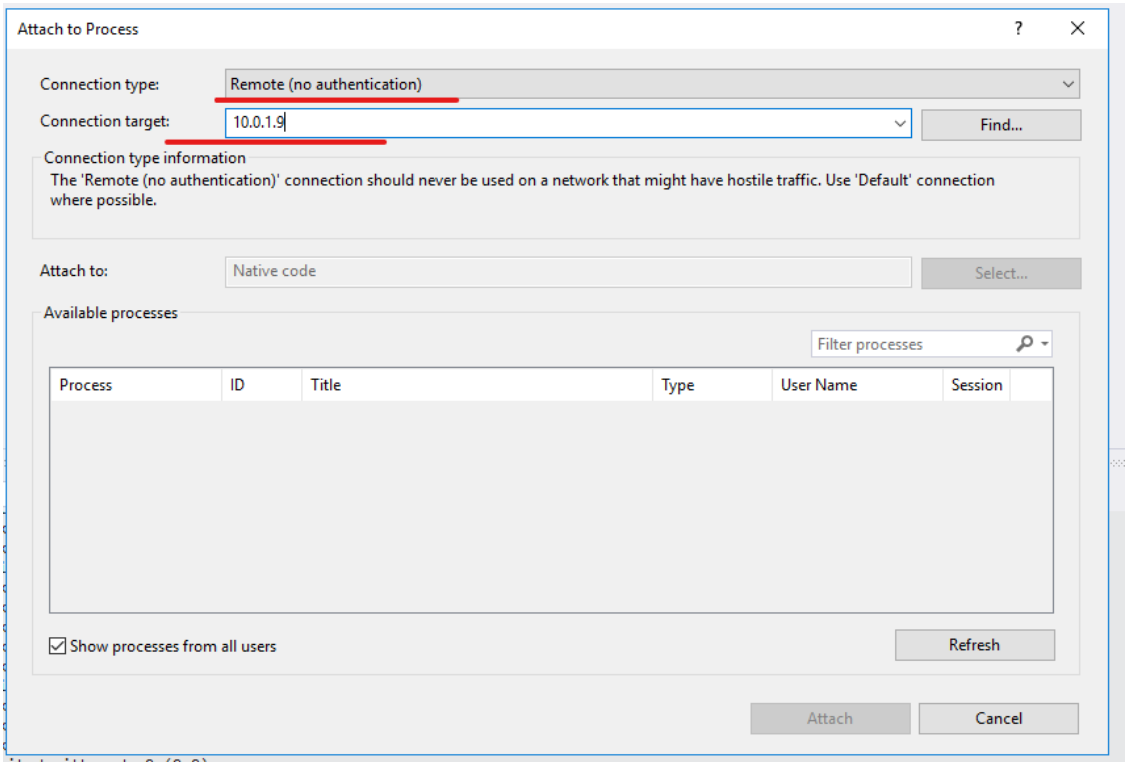
6.3 Debugging

On the local machine where plugins or modules were created, users just need to select the process in the Visual Studio. The Remote Debugger will be attached to the selected process. For that:

b) Launch Visual Studio.
c) Click the **Debug** section on the menu, and select the **Attach to Process…** menu item.



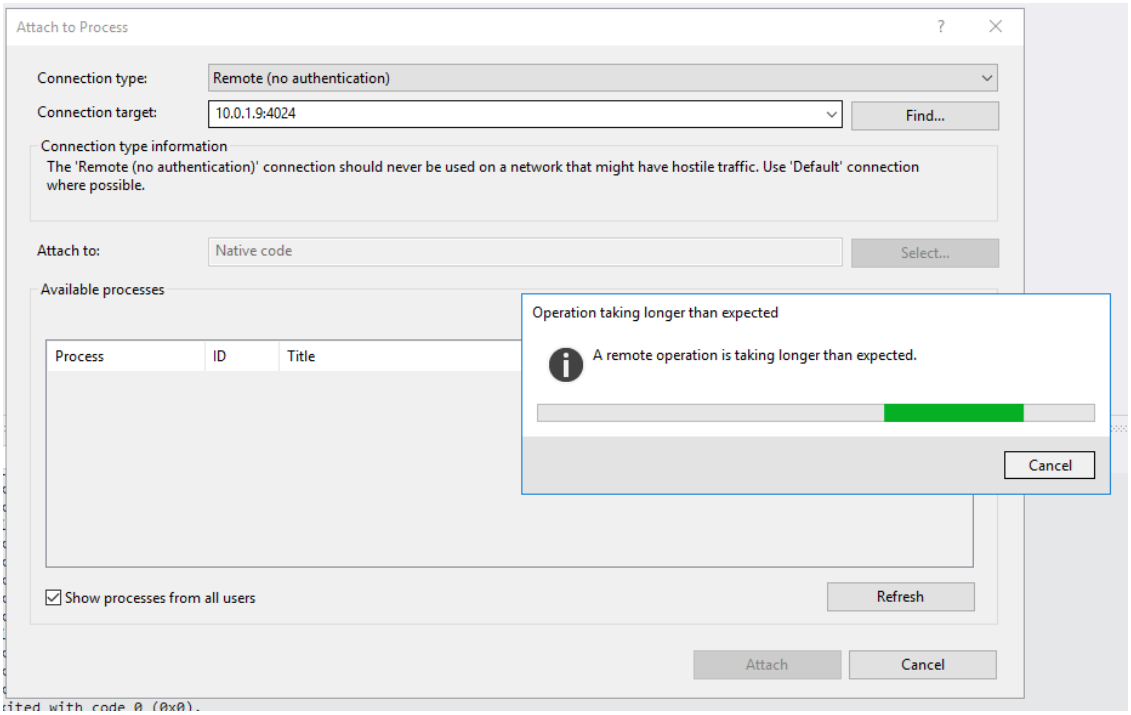
The **Attach to Process** pop-up window opens.



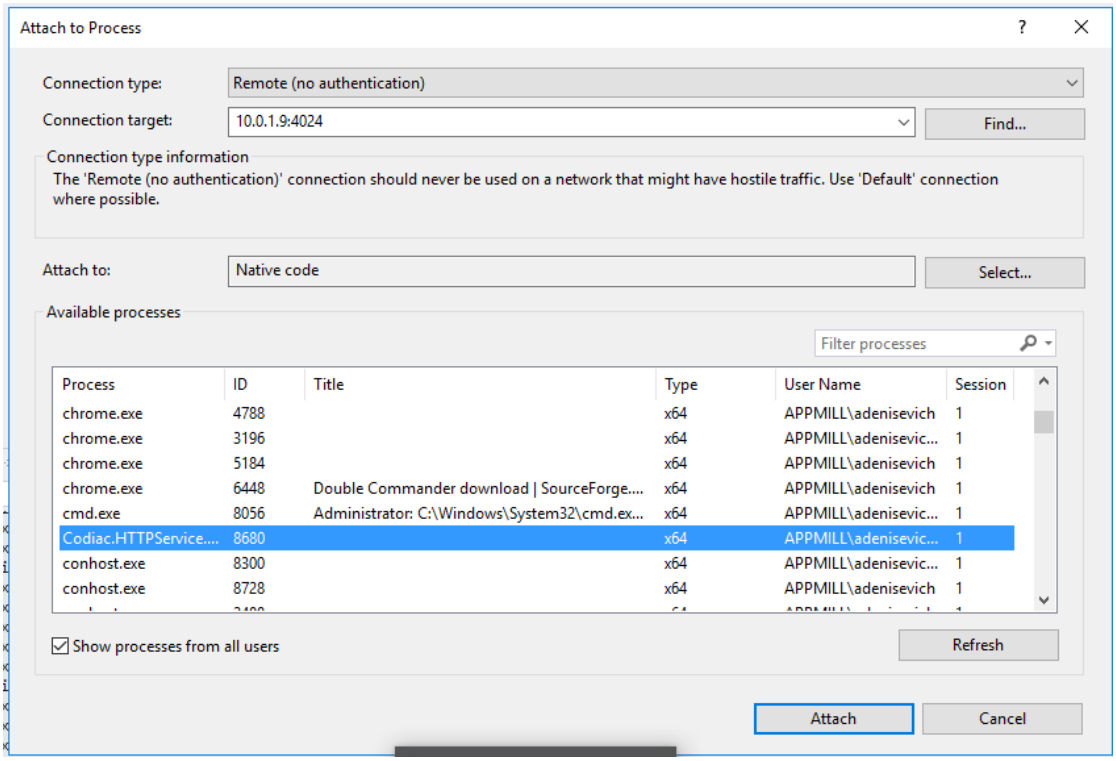
d) In the opened **Attach to Process** pop-up window, select:
   - the **Remote (No authentication)** debugger type from the Connection type drop-down list,
   - the remote machine IP address from the Connection target drop-down list, and

- press the ENTER key.

The debugger will connect to the remote machine, and the list of available processes that the debugger can be attached to will be shown.



e) In the available processes list, select the **Codiac.HTTPService.exe** process, and click the **Attach** button.



At reaching a code for the plugins, the process will be stopped at the breakpoint that was set in the developed plugin or modules.

To allow the remote debugging, files with the Debug information that were generated at the build should be added.

7 C++ helper functions

7.1 Helper functions for C++ AF extension function

These helper functions are macros for directly accessing the aliases of Alias Framework. So, the AF extension functions can directly update the evaluated aliases.

7.1.1 GET_FROM_MAP(*alias_name*)

- **Purpose:** return AliasValue object of an alias
- **Example:** This example returns the AliasValue object of "Alias.EmailSendTests.language". With the AliasValue object, the string value can be retrieve using valueToString method.

```
IAliasValuePtr& avLanguage = GET_FROM_MAP(TEXT("Alias.EmailSendTests.language"));
const utility::string_t language = avLanguage->valueToString();
if (language == TEXT("C++"))
{
```

7.1.2 GET_FROM_MULTIMAP(alias_name)

- **Purpose:** return array of AliasValue object of multi-value alias
- **Example:** This example returns the array of AliasValue objects of "Array.CrunchF.Amt".Then, iterate on that vector of AliasValue objects for processing each item.

```
auto avAmounts = GET_FROM_MULTI_MAP(TEXT("Array.CrunchF.Amt"));

double total = 0;
for (auto& avAmount : avAmounts)
{
    utility::string_t amount = avAmount->valueToString();
```

7.1.3 INSERT_IN_MAP(key, value)

7.1.4 REPLACE_OR_INSERT_IN_MAP(key, value)

- **Purpose:** insert alias (with AliasValue object) in evaluated aliases
- **Example:**This example creates an AliasValue object and add it in evaluated aliases.

```
IAliasValuePtr newAliasValue = codiacsdk_api::CreateAliasValue();
newAliasValue->setValue(total);
REPLACE_OR_INSERT_IN_MAP(TEXT("Custom.ExampleModule.EmailCrunchFTotal"), newAliasValue);
```

- Note: The INSERT_IN_MAP is similar to REPLACE_OR_INSERT_IN_MAP except that it doesn't insert if the Alias name exist in evaluated aliases.

7.1.5 INSERT_IN_MULTI_MAP(key, value)

7.1.6 REPLACE_OR_INSERT_IN_MULTI_MAP(key, value)

- **Purpose:** insert multi-value Alias (with vector of AliasValue objects) in evaluated aliases which
- **Example:**This example creates vector of AliasValue objects and add it in evaluated aliases.

```
std::vector<IAliasValuePtr> outputAlias;
for (auto iter = fetchedSql.begin(); iter != fetchedSql.end(); iter++)
{
    IAliasValuePtr av = codiacsdk_api::CreateAliasValue();
    av->setValue(iter->c_str());
    outputAlias.push_back(av);
}

INSERT_IN_MULTI_MAP(TEXT("Multi.ExampleModule.SqlSelectOtherDb"), outputAlias);
```

- **Note:** The INSERT_IN_MAP is similar to REPLACE_OR_INSERT_IN_MAP except that it doesn't insert if the Alias name exist in evaluated aliases.

7.2 AliasValue class

7.2.1 AliasValue constructor

- **Purpose:** create AliasValue object

7.2.2 setValue(value)

- **Purpose:** set AliasValue object with double value, integer, or string

7.2.3 valueToInt(), valueToDouble(), valueToString()

- **Purpose:** convert the Alias value to integer, to double or return the alias string.

7.2.4 IsValueNull()

- **Purpose:** check if the Alias value is uninitialized

7.2.5 markForDeletion()

- **Purpose:** delete database table row pointed by Alias value after execution of AF extension function

7.2.6 isMarkForDeletion()

- **Purpose:** check if the Alias value is for deletion

Note: See the helper functions for AF for example usage of AliasValue class.

7.3 AliasView class for C++ plugins

7.3.1 codiacsdk_api::CreateNewAliasView(aliasView, aliasInput,

ruleInput, userInformation, dataPoolLoader) function

- **Purpose:** create AliasView instance that evaluates alias view

7.3.2 Example

- This example creates and AliasView instance and evaluates “CrunchView” alias view with input “Alias.Crunch.Contract” alias having value of “SAMPLEVUL”. The evaluate method stored the evaluated data to the evaluatedData argument.

```
web::json::value aliasViews;
aliasViews[0] = web::json::value(TEXT("CrunchView"));

web::json::value aliasInput;
aliasInput[TEXT("Alias.Crunch.Contract")] = web::json::value(TEXT("SAMPLEVUL"));

// create AliasView object
auto av = codiacsdk_api::CreateAliasView(aliasViews,
    aliasInput,
    funcParam,
    funcParam,
    this->poolLoader.Get(),
    codiacsdk_api::GetJsonKeeper());

// evaluate CrunchView alias view(alias framework)
web::json::value evaluatedData;
bool status = av->evaluate(evaluatedData);
```

7.4 AliasFramework helper functions for C++ plugins

7.4.1 codiacsdk_api::_GetAFRequest() function

- **Purpose:** creates AF request that contain alias input and alias output to be evaluated; and output data (or evaluated aliases).
    - setAliasInput, setAliasOutput, getOutData

7.4.2 codiacsdk_api::_GetAFClass(userInformation, dataPoolLoader) function

- **Purpose:** Create AF instance that evaluates aliases
- **Example:** This example create AF instance and evaluates the input and output aliases. It is done by creating the AF request and AF instance. Then, the input aliases JSON and the output aliases JSON are assigned to AF request which is passed to AF evaluate method. Finally, the evaluate method stores the evaluated aliases in outData of AF request.

```
auto req = codiacsdk_api::_GetAFRequest();

req->setNumberOfThreads(5);
req->setRequestPrimaryTable(TEXT("Crunch"));

web::json::value inputContract;
inputContract[TEXT("pk")] = web::json::value(TEXT("SAMPLEVUL"));
inputContract[TEXT("value")] = web::json::value(TEXT("SAMPLEVUL"));

web::json::value inputOwner;
inputOwner[TEXT("pk")] = web::json::value(TEXT("SAMPLEVUL"));
inputOwner[TEXT("value")] = web::json::value(TEXT("Test Owner"));

web::json::value aliasInput;
aliasInput[TEXT("Alias.Crunch.Contract")] = inputContract;
aliasInput[TEXT("Alias.Crunch.Owner")] = inputOwner;
req->setAliasInput(aliasInput.serialize().c_str());

web::json::value outputs;
outputs[0] = web::json::value(TEXT("Alias.Crunch.Owner"));
outputs[1] = web::json::value(TEXT("Alias.Crunch.MaturityDate"));
req->setAliasOutput(outputs.serialize().c_str());

auto af = codiacsdk_api::_GetAFClass(funcParam.serialize().c_str(), &(this->poolLoader));

af->evaluate(req);
auto outData = req->getOutData();
```

7.5 JSON request helper functions for C++ plugins

These helper functions parse the JSON request and provide convenient way to retrieve and update the values of aliases. This is are normally use for plugins that going to be assigned as pre-extension functions.

7.5.1 codiacsdk_api::GetAliasValue(alias)

- **Purpose:** retrieve the string value of an alias
- **Example:** This example retrieve the value of “Alias.Crunch.Contract” alias from JSON request.

```
utility::string_t contract = codiacsdk_api::GetAliasValue(TEXT("Alias.Crunch.Contract"));
```

7.5.2 codiacsdk_api::SetAliasValue()

- **Purpose:** modify the value of an alias
- **Example:** This example modifies “Alias.Crunch.Notes” alias of the JSON request to “Updated by plugin”.

```
codiacsdk_api::SetAliasValue(TEXT("Alias.Crunch.Notes"), TEXT("Updated by plugin"));
```

7.5.3 codiacsdk_api::GetAliasArraySize()

- **Purpose:** returns the size of alias array
- **Note:** See the example of GetAliasArrayValue for usage

7.5.4 codiacsdk_api::GetAliasArrayValue(index, alias)

- **Purpose:** retrieve the string value of alias from an array index
- **Example:** This example retrieves the value of each "Array.CrunchF.Amt" alias from JSON request.

```
long numRowsForUpdate = codiacsdk_api::GetAliasArraySize();
for (long i = 0; i < numRowsForUpdate; i++)
{
    // get Alias values by array index
    utility::string_t amount = codiacsdk_api::GetAliasArrayValue(i, TEXT("Array.CrunchF.Amt"));
```

7.5.5 codiacsdk_api::SetAliasArrayValue(index, alias, value)

- **Purpose:** modify the value of an alias from an array index
- **Example:** This example replaces the value of "Array.CrunchF.Amt" with the value of newAmount variable.

```
codiacsdk_api::SetAliasArrayValue(i, TEXT("Array.CrunchF.Amt"), newAmount);
```

7.5.6 codiacsdk_api::GetAliases()

- **Purpose:** return aliases (excludes alias array) from values of aliases of JSON request (in map of AliasValue objects)
- **Example:** This example iterate on map of aliases and prints the each values

```
ObjectPtr<IAliasValuesMap> aliases = codiacsdk_api::GetAliases();
size_t size = aliases->size();
for (size_t index = 0; index < size; index++)
{
    auto pair = aliases->at(index);
    utility::string_t aliasName = pair->key();
    auto aliasValue = pair->value();

    if (pluginsLogs)
        std::wcout << TEXT("Alias: ") << aliasName
            << TEXT(" Value: ") << aliasValue->valueToString()
            << std::endl;
}
```

7.5.7 codiacsdk_api::GetAliasArray()

- **Purpose:** return entire alias array of JSON request (in map of vector of AliasValue objects)
- **Example:** This example iterate on map of aliases then iterate on vector of AliasValue objects. Then, print the value of aliases. The GetAliasArray() function is another way to get

the values of alias array in addition to GetAliasArrayValue() function.

```
ObjectPtr<IAliasDataMV> aliasArray = codiacsdk_api::GetAliasArray();

aliasArray->ForEach([&](IMapPairPtr<IString*, IVectorPtr<IAliasValuePtr>> pair)
{
    IString* aliasName = pair->first();

    if (pluginsLogs)
        std::wcout << TEXT("Alias ") << aliasName->str() << std::endl;

    IVectorPtr<IAliasValuePtr> aliasvalueVector = pair->second();

    for (size_t index = 0; index < aliasvalueVector->size(); index++)
    {
        if (pluginsLogs)
            std::wcout << TEXT(" has ") << aliasvalueVector->at(index)->valueToString() << std::endl;
    }
});
```

7.6 Message helper functions for C++ plugins

7.6.1 codiacsdk_api::IsConfirmed(messageCode)

- **Purpose:** Check if the JSON request has confirmed messageCode. This helper function is use with plugin_helpers::Info and plugin_helpers::Warning.
- Note: See the examples of plugin_helpers::Info and plugin_helpers::Warning for usage of IsConfirmed function.

7.6.2 codiacsdk_api::Info(messageCode, message)

- **Purpose:** JSON response with Info message
- **Example:** This example sends back a message response to client with an id of 1.

```
if (!codiacsdk_api::IsConfirmed(1))
{
    utility::string_t response = codiacsdk_api::Info(1, TEXT("Starting a task"));
    return CloneString(response.c_str());
}
```

- Client output:

- Note: The client will resend the request after it has acknowledge the Info message. So, the

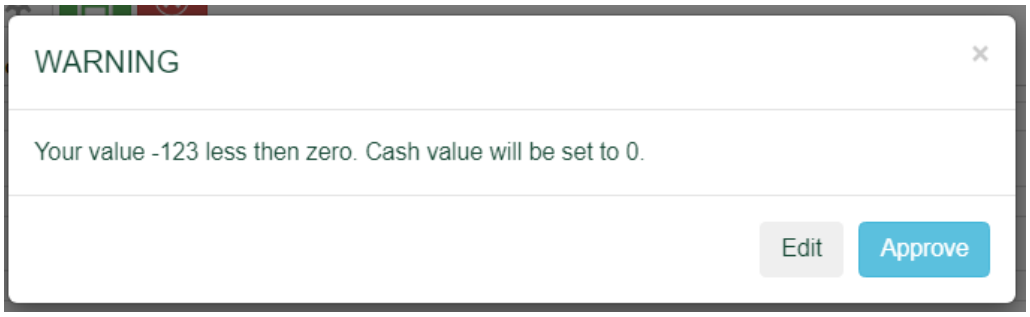
plugin function must check the messageCode to skip the Info message.

7.6.3 codiacsdk_api::Warning(messageCode, message)

- **Purpose:** JSON response with Warning message
- **Example:** This example sends back a warning message response to client with an id of 2.

```
if (IsConfirmed(2))
{
    cash = 0;
}
else
{
    utility::string_t message = TEXT("Your value ");
    message += cashValue;
    message += TEXT(" less then zero. Cash value will be set to 0.");
    utility::string_t response = Warning(2, message.c_str());
    return CloneString(response.c_str());
}
```

- **Client output:**



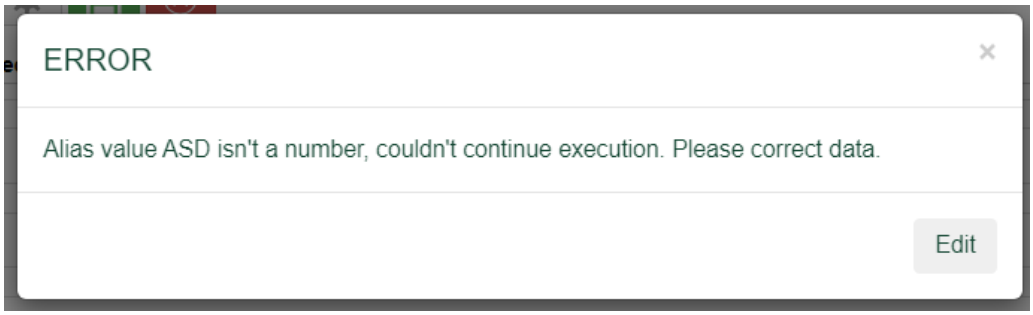
- **Note:** The client will resend the request after it has acknowledge the Warning message. So, the plugin function must check the messageCode to skip the Warning message.

7.6.4 codiacsdk_api::Error(messageCode, message)

- **Purpose:** JSON response with Error message
- **Example:** This example sends back an error message response to client".

```
// if invalid number, send message to client that value is not a number
utility::string_t message = TEXT("Alias value ");
message += cashValue;
message += TEXT(" isn't a number, couldn't continue execution. Please correct data.");
utility::string_t response = Error(4, message.c_str());
return CloneString(response.c_str());
```

- **Client output:**



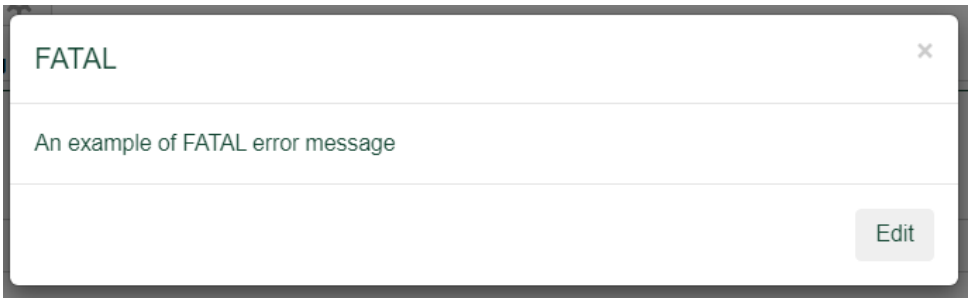
- **Note:** The client doesn't resend the request after it has acknowledge the Error message.

7.6.5 codiacsdk_api::Fatal(messageCode, message)

- **Purpose:** JSON response with Fatal message
- **Example:** This example sends back an error message response to client.

```
utility::string_t response = Fatal(MessagesEnum::FATAL_CODE, TEXT("An example of FATAL error message"));
return CloneString(response.c_str());
```

- **Client output:**



- **Note:** The client doesn't resend the request after it has acknowledge the Error message.

7.7 Document management helper functions for C++ plugins

7.7.1 codiacsdk_api::CopyDocumentToDevice(filePath, fileDescription, documentFamily, documentCategory)

- **Purpose:** Copies document to a specific device, the family and category are optional parameters, if they are not specified the document is copied according to the parameters in the JsonKeeper.
- **Return:** Returns the file container ID if successful, otherwise an empty string is returned.
- **Note:** The example for this function is provided together with the "AddDocumentToReturnList" function example.

7.7.2 codiacsdk_api::AddDocumentToReturnList(documentID)

- **Purpose:** Adds the file container ID to the JsonKeeper and later in the response of the extension function.
- **Return:** Returns “true” if successfully executed, otherwise “false” is returned.
- **Example:**

```
bool isAdded = false;
utility::string_t fileID = TEXT("");
utility::string_t filePath = TEXT("C:\\DocumentManagementTest.txt");
utility::string_t fileDescription = TEXT("description");

utility::string_t documentFamily = TEXT("test");
utility::string_t documentCategory = TEXT("test");

// copies the document to the device specified on the screen
fileID = codiacsdk_api::CopyDocumentToDevice(filePath, fileDescription);
// adds the file container ID in the response
isAdded = codiacsdk_api::AddDocumentToReturnList(fileID);
std::wcout << TEXT("File container ID: ") << fileID << TEXT(". Added to return list: ") << isAdded << std::endl;

// copies the document to the device specified in the parameters
fileID = codiacsdk_api::CopyDocumentToDevice(filePath, fileDescription, documentFamily, documentCategory);
// adds the file container ID in the response
isAdded = codiacsdk_api::AddDocumentToReturnList(fileID);
std::wcout << TEXT("File container ID: ") << fileID << TEXT(". Added to return list: ") << isAdded << std::endl;

return PrepareResponse(TEXT("OK"), true);
```

7.7.3 codiacsdk_api::GetDocumentFromDevice(fileContainerID, filePath)

- **Purpose:** Loads the document with the specified file container ID in the specified path.
- **Return:** Returns “true” if successfully executed, otherwise “false” is returned.
- **Example:**

```
bool isLoaded = false;
utility::string_t fileID = TEXT("1111");
utility::string_t filePath = TEXT("D:\\Users\\spoposki\\Desktop\\testFile.jpg");

// loads the document with the specified file container ID in the specified path
isLoaded = codiacsdk_api::GetDocumentFromDevice(fileID, filePath);
std::wcout << TEXT("File container ID: ") << fileID << TEXT(". Is loaded: ") << isLoaded << TEXT(". Location: ") << filePath << std::endl;

return PrepareResponse(TEXT("OK"), true);
```

7.7.4 codiacsdk_api::DeleteDocumentFromDevice(fileContainerID, performHardDelete)

- **Purpose:** Sets the file container “Active” flag to “false” or deletes the file container completely, it depends on the value of the “performHardDelete” parameter.
- **Returns:** Returns “true” if successfully executed, otherwise “false” is returned.
- **Example:**

```
bool isDeleted = false;
utility::string_t fileID = TEXT("1111");

// performs a soft delete on the specified file container
isDeleted = codiacsdk_api::DeleteDocumentFromDevice(fileID, false);
std::wcout << TEXT("File container ID: ") << fileID << TEXT(". Is soft deleted: ") << isDeleted << std::endl;

// performs a complete delete on the specified file container
isDeleted = codiacsdk_api::DeleteDocumentFromDevice(fileID, true);
std::wcout << TEXT("File container ID: ") << fileID << TEXT(". Is completely deleted: ") << isDeleted << std::endl;

return PrepareResponse(TEXT("OK"), true);
```

7.7.5 codiacsdk_api::SearchForDocument(documentFamily, documentCategory, kp1, kp2, kp3, kp4, kp5, description)

- **Purpose:** Searches for file containers using the function parameters as search parameters. If a function parameter is an empty string that field will not be included in the search.
- **Return:** Returns serialized JSON array containing information about the file containers that match the search parameters.
- **Example:**

```
utility::string_t fileRootPath = TEXT("D:\\Users\\spoposki\\Desktop\\");
utility::string_t description = TEXT("test1");

// searches for the documents that have the value "test" in description
utility::string_t files = codiacsdk_api::SearchForDocument(TEXT(""), TEXT(""), TEXT(""), TEXT(""), TEXT(""), TEXT(""), TEXT(""), description);
std::wcout << TEXT("Files: ") << files << std::endl;
// parses the response and iterates through the array
jvalue filesJson = jvalue::parse(files);
cr_jarray filesArray = filesJson.as_array();
for(auto file : filesArray)
{
    bool isLoaded = false;
    utility::string_t filePath = fileRootPath + file.at(TEXT("container_file_name")).as_string();
    // loads the document to disk
    isLoaded = codiacsdk_api::GetDocumentFromDevice(std::to_wstring(file.at(TEXT("id")).as_integer()), filePath);
    std::wcout << TEXT("File container ID: ") << file.at(TEXT("id")).as_integer() << TEXT(". Is loaded: ") << isLoaded << TEXT(". Location: ") << filePath << std::endl;
}

return PrepareResponse(TEXT("OK"), true);
```

7.7.6 codiacsdk_api::SetDocumentKeyParts(fileContainerID, kp1, kp2, kp3, kp4, kp5)

- **Purpose:** Sets the file container key parts.
- **Return:** Returns “true” if successfully executed, otherwise “false” is returned.
- **Example:**

```
bool isSet = false;

utility::string_t fileID = TEXT("1111");
utility::string_t kp1 = TEXT("test_kp1");
utility::string_t kp2 = TEXT("test_kp2");
utility::string_t kp3 = TEXT("test_kp3");
utility::string_t kp4 = TEXT("test_kp4");
utility::string_t kp5 = TEXT("test_kp5");

// sets the key parts
isSet = codiacsdk_api::SetDocumentKeyParts(fileID, kp1, kp2, kp3, kp4, kp5);
std::wcout << TEXT("Key parts set : ") << isSet;

return PrepareResponse(TEXT("OK"), true);
```

7.8 Database helper functions for C++ plugins and C++ AF extension functions

7.8.1 codiacsdk_api::DBCursor

- **Purpose:** store the query results of ExecuteSQLHelper; and able to fetch row sequentially from that query results
- **Methods:**
  - Size() returns number of rows
  - SetPosition(position) set the position of cursor in query results
  - GetPosition() return the position of cursor in query results
  - FetchOne() returns row (in vector of columns); and next FetchOne() call returns next row
  - GetColumnNames() returns the column names of query results. The column names are in alphabetic order.
- **Note:** See the example of ExecuteSQLHelper function for usage

7.8.2 codiacsdk_api::ExecuteSQLHelper(dbCursor, sqlQuery, dbType, dbAddress, dbName, dbUser, dbPass, dbPort)

- **Purpose:** query the database by running SQL query to Codiac database or to other database (with connection information)
- **Note:** dbType, dbAddress, dbName, dbUser, dbPass and dbPort parameters are optional and default to empty which means to connect to Codiac database
- **Example:** This example run SQL query to Codiac database and store the first column to vector of AliasValue objects.

```
ObjectPtr<IDBCursor> sqlResult = executeSql(
    TEXT("SELECT \"Crunch\".\"Contract\" FROM \"Crunch\"")
);

std::vector<IAliasValuePtr> outputAlias;
for (int i = 0; i < sqlResult->Size(); i++)
{
    std::vector<utility::string_t> fetchedSql = sqlResult->FetchOne();
    IAliasValuePtr av = codiacsdk_api::CreateAliasValue();
    av->setValue(fetchedSql[0].c_str());
    outputAlias.push_back(av);
}
```

- **Example:** This example run SQL query to postgres database (with connection information).

```
DBConnector dbConnector;
dbConnector.SetDBType(TEXT("Postgres"));
dbConnector.SetDBName(TEXT("postgres"));
dbConnector.SetDBUser(TEXT("postgres"));
dbConnector.SetDBPass(TEXT("champ123"));
dbConnector.SetDBPort(TEXT("5432"));
dbConnector.SetDBAddress(TEXT("192.168.100.204"));

ObjectPtr<IDBCursor> sqlResult = executeSql(
    TEXT("SELECT \"Crunch\".\"Contract\" FROM \"Crunch\""),
    dbConnector
);
```

7.9 Other CodiacSDK helper functions for C++ plugins and C++ AF extension functions

7.9.1 codiacsdk_api::SendEmail(to, subject, body)

- **Purpose:** send an email to recipient with subject and message
- **Example:** This AF extension function example sends an email with recipient from “Alias.EmailSendTests.recipient” alias, subject from “Alias.EmailSendTests.subject” alias and message from “Alias.EmailSendTests.body” alias.

```
IAliasValuePtr& avRecipient = GET_FROM_MAP(TEXT("Alias.EmailSendTests.recipient"));
const utility::string_t recipient = avRecipient->valueToString();
IAliasValuePtr& avSubject = GET_FROM_MAP(TEXT("Alias.EmailSendTests.subject"));
const utility::string_t subject = avSubject->valueToString();
IAliasValuePtr& avBody = GET_FROM_MAP(TEXT("Alias.EmailSendTests.body"));
const utility::string_t body = avBody->valueToString();
codiacsdk_api::SendEmail(recipient, subject, body);
```

7.9.2 codiacsdk_api::SendEmail(to, subject, body, attachments)

- **Purpose:** Sends an email to recipient with subject, message and attachments. The attachment parameter is a delimited string of file container IDs.
- **Return:** A serialized JSON with details.
- **Example:**

```
utility::string_t to = TEXT("slavkopoposki@gmail.com");
utility::string_t subject = TEXT("SUBJECT");
utility::string_t body = TEXT("BODY");

utility::string_t files = TEXT("1178;1179;1180");

// sends an email with attachments, the file that correspond to the id will the attached to the email
codiacsdk_api::SendEmail(to, subject, body, files);

return PrepareResponse(TEXT("OK"), true);
```

7.9.3 codiacsdk_api::SendSms(to, body)

- **Purpose:** send SMS message to recipient with message
- **Example:** This example send SMS message to 5556789090. Only US numbers are accepted.

```
codiacsdk_api::SendSms(TEXT("5556789090"), TEXT("This is test message"));
```

7.9.4 codiacsdk_api::GenerateReport(reportName, jsonData, isSaveToXml)

- **Purpose:** generate report PDF file from report template (client report builder)
- **Note:** The isSaveToXml default to false; if set to true, the report are save to XML file and is not generated to PDF.
- **Example:** This example generate report from alias view data. The alias view data contains alias values in JSON format.

```
web::json::value parsedInput = web::json::value::parse(pJsonParams);
web::json::value aliasViewData = parsedInput.at(TEXT("alias_view_data"));
utility::string_t inlineJson = (aliasViewData.is_array() && aliasViewData.size() > 0
    ? aliasViewData[0].serialize()
    : aliasViewData.serialize());

utility::string_t reportName = TEXT("Test CrunchF Report");

utility::string_t response = codiacsdk_api::GenerateReport(reportName, inlineJson);
return CloneString(response.c_str());
```

- **Example:** Another example that generate report from report template “Test CrunchF Report” with values from dependent multi-value Aliases. Note that the function has to create

a JSON input string from aliases.

```
// dependency aliases
auto contracts = GET_FROM_MULTI_MAP(TEXT("Array.CrunchF.Contract"));
auto descriptions = GET_FROM_MULTI_MAP(TEXT("Array.CrunchF.Description"));
auto fundNos = GET_FROM_MULTI_MAP(TEXT("Array.CrunchF.FundNo"));
auto amounts = GET_FROM_MULTI_MAP(TEXT("Array.CrunchF.Amt"));

// list of array aliases in json
web::json::value jsonInput;
for (size_t i = 0; i < contracts.size(); i++)
{
    web::json::value jsonCrunchF;
    jsonCrunchF[TEXT("Array.CrunchF.Contract")] = web::json::value(contracts[i]->valueToString());
    jsonCrunchF[TEXT("Array.CrunchF.Description")] = web::json::value(descriptions[i]->valueToString());
    jsonCrunchF[TEXT("Array.CrunchF.FundNo")] = web::json::value(fundNos[i]->valueToString());
    jsonCrunchF[TEXT("Array.CrunchF.Amt")] = web::json::value(amounts[i]->valueToString());
    jsonInput[TEXT("Array.CrunchF")][i] = jsonCrunchF;
}

// generate report
utility::string_t result =codiacsdk_api::GenerateReport(TEXT("Test CrunchF Report"), jsonInput.serialize(), false);
```

- Another **Example:** This example generate report from report template “Test CrunchF Report” with values from vector of delimited values. The delimiter used '|'.

```
const std::vector<utility::string_t> inlineData{
    TEXT("Contract|FundNo|Description|Amount"),
    TEXT("Contract A|CA1|Test contract A - 1|123.45"),
    TEXT("Contract A|CA2|Test contract A - 2|3.45"),
    TEXT("Contract B|CB1|Test contract B - 1|645"),
    TEXT("Contract C|CC1|Test contract C - 1|7.89"),
};

utility::string_t result = codiacsdk_api::GenerateReport(TEXT("Test CrunchF Report - inline data"),
    inlineData, TEXT("|"));
```

7.9.5 codiacsdk_api::ExecuteJson(jsonRequest)

- **Purpose:** call Codiac service by executing JSON request
- **Example:** This example calls JobsScheduler::GetJobsScheduleList to get list of jobs. A JSON request is created with func_name parameter and lib_name parameter.

```
// create json request
web::json::value jsonRequestBody;
jsonRequestBody[TEXT("func_name")] = web::json::value(TEXT("GetJobScheduleList"));
jsonRequestBody[TEXT("lib_name")] = web::json::value(TEXT("CodiacSDK.zJobsScheduler"));

web::json::value jsonRequest;
jsonRequest[TEXT("requestbody")] = jsonRequestBody;

// execute service to get Jobs
ObjectPtr<ICodiacJson> response = codiacsdk_api::ExecuteJson(jsonRequest.serialize().c_str());
```

- **Note:** The user\session parameters are filled up at service process.

7.10 SQL query with C++ native database libraries

Codiac SDK provides database helper functions that can access Codiac database and other databases with connection string. In addition, the user can directly call the C++ native library to query the database. The Codiac ~external directory contains the C++ native libraries for PostgreSQL, MySQL, MS SQL Server, and Oracle SQL.

The Codiac.Modules\ExampleModule\SqlExamples.cpp and CodiacSDK.DirectSqlExamples project contain example source code on how to query the databases with native libraries.

- PostgreSQL

  The example source codes uses the libpqxx library to query PostgreSQL database. Below is the sequence for libpqxx to query database:

  1. Connect to database with connection string using “pqxx::connection”.
  2. Create a transaction with “pqxx::work” object and execute query with “exec” on that transaction.
  3. Commit that transaction with “pqxx::commit”.
  4. A non-transaction mode can also be use with “pqxx::nontransaction” object. This mode doesn't need to be committed. Execute a query with “exec” with non-transaction object.
  5. Finally, “disconnect” on the connection object.

  See the PostgreSQL function in SqlExample.cpp and postgreSql function of CodiacSDK.DirectSqlExamples.cpp for full example.

  To build plugin or module with libpqxx, you will need libpqxx development package. See https://github.com/jtv/libpqxx on how to build libpqxx. After building the libpqxx, set the CMakeLists.txt's LIBPQXX_INCLUDE_DIR to where the libpqxx header files are and LIBPQXX_LIB_DIR to where the libqxx libraries are.

  CMakeLists.txt

```
50
51  IF(NOT DEFINED LIBPQXX_INCLUDE_DIR)
52      SET(LIBPQXX_INCLUDE_DIR ${EXTERNAL_LIB_DIR}/libpqxx/include)
53  ENDIF()
54  IF(NOT DEFINED LIBPQXX_LIB_DIR)
55      SET(LIBPQXX_LIB_DIR ${EXTERNAL_LIB_DIR}/libpqxx/lib)
56  ENDIF()
```

  See the http://pqxx.org/development/libpqxx for full details on how to use libpqxx.

- MySQL

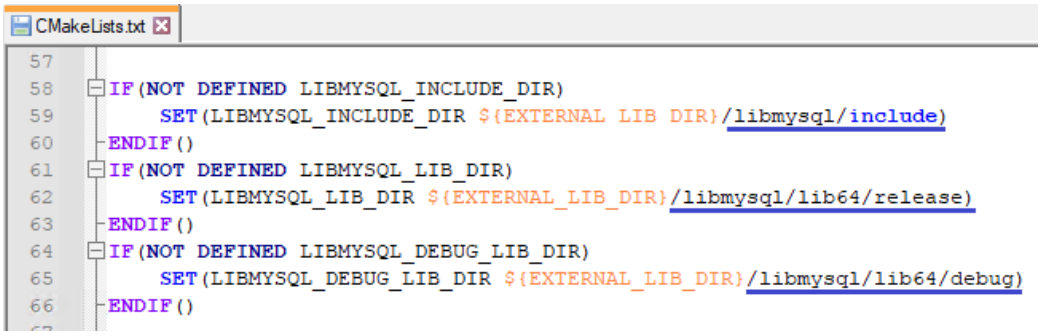
  The example source code uses MySQL Connector C++ to query MySQL database. Below is the sequence for MySQL Connector to query database:

  1. Create a connection object with connect to driver instance
  2. Specify the database with setSchema on connection object

3. Create a statement object for SQL query on connection object
4. Execute a query with “execute” on statement object.
5. For query that return result set, query with “executeQuery” on statement object.
6. Free up the created objects.

See the MySQL function of SqlExamples.cpp and mySql function of CodiacSDK.DirectSqlExamples.cpp for full example.

To build plugin or module with MySQL Connector C++, you will need to run MySQL installer (see https://dev.mysql.com/doc/connector-cpp/8.0/en/connector-cpp-installation-binary.html). After installation, set the CMakeLists.txt's LIBMYSQL_INCLUDE_DIR to where the MySQL Connector header files are and LIBMYSQL_LIB_DIR to where the MySQL Connector libraries are.



See the https://dev.mysql.com/doc/connector-cpp/8.0/en/ for full details on how to use MySQL Connector.

- MS SQL Server

  The example source code uses Microsoft ODBC driver for SQL Server. Below is the sequence for Microsoft ODBC driver to query database:

  1. Allocate environment handle with SQLAllocHandle
  2. Allocate DB connection handle with SQLAllocHandle
  3. Connect to database driver with connection string with SQLDriverConnect
  4. Allocate statement handle with SQLAllocHandle
  5. Execute query with SQLExecDirect with statement handle
  6. Fetch query results with SQLFetch with statement handle
  7. Retrieve each column result with SQLGetData with statement handle
  8. Free up statement handle with SQLFreeStmt for every call to SQLExecDirect
  9. Free up DB connection handle
  10. Free up environment handle

  See the MsSQLServer function of SqlExamples.cpp and MsSqlExmple.cpp for full example.

  See the https://docs.microsoft.com/en-us/sql/connect/odbc/microsoft-odbc-driver-for-sql-

server for full details on how to install and use MS ODBC driver for MS SQL Server.

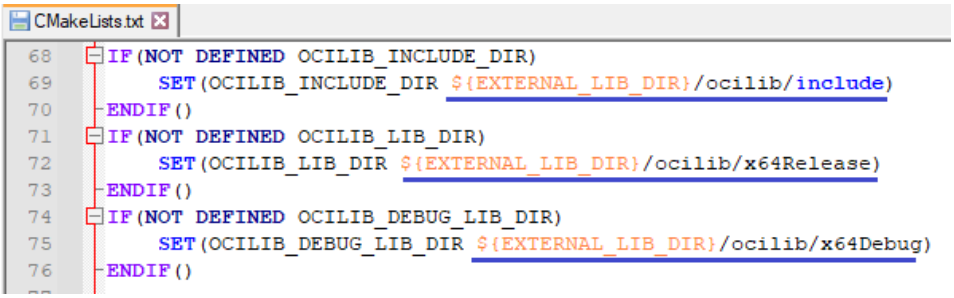
- Oracle SQL

  The example source code uses OCILIB driver for Oracle. Below is the sequence for OCILIB driver to query database:

  1. Call “ocilib::Environment::Iniatilize” function to initialize OCI library .
  2. Create connection object (“ocilib::Connection”) with database connection information.
  3. Create statement object (“ocilib::Statement”) with connection object
  4. Call “Execute” function on statement object to execute a query
  5. Call “GetResultset” function on statement object to get the query result set object.
  6. Call “Get” function on result set object to retrieve the column value.
  7. Free up ocilib objects

  See the OracleSQL function of SqlExamples.cpp and oracleSql function of CodiacSDK.DirectSqlExamples.cpp for full example.

  To build plugin or module with OCILIB, you will need to download and install OCILIB (see https://vrogier.github.io/ocilib/download). After installation, set the CMakeLists.txt's OCILIB_INCLUDE_DIR to where the OCILIB header files are and OCILIB_LIB_DIR and OCILIB_DEBUG_LIB_DIR to where the OCILIB libraries are.



  See the https://vrogier.github.io/ocilib/ for full details on how to use OCILIB driver for Oracle.

8 Python Alias Framework extension module

The Alias Framework (AF) service can evaluate custom generated aliases or multi generated aliases. This is done by going through the alias name structure:

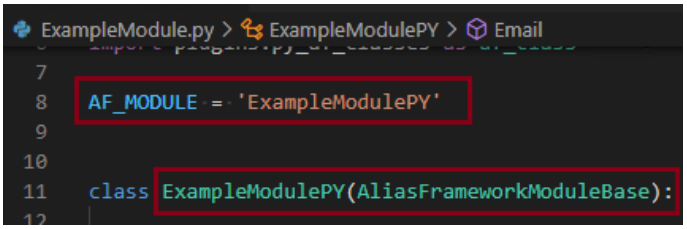
AliasType.ModuleName.FunctionName

Depending on the *AliasModuleType*, the Custom\Multi Generated alias can be C++ function. Python function, JavaScript function or PHP function.

For example, the “Custom.ExampleModulePY.CalculateAverage” tells us that the:

- alias type is custom generated alias,
- if the AliasModuleType is Python, the Python class name is ExampleModulePY,
- and python function name is CalculateAverage.

Then, the AF looks in `<service directory>\plugins\python` directory for Python scripts with AF_MODULE variable. This tells AF that the Python script is an AF extension module. For example, the image below refers to ExampleModule.py that contains an AF extension module with a class name of ExampleModulePY.



Then, AF loads Python script with ExampleModulePY class and executes CalculateAverage function which perform some tasks or calculations.

8.1 Required tools

- Visual Studio Code - https://code.visualstudio.com/
- Visual Studio Code: Python extension by Microsoft

The Codiac.Service\~external directory contains Python 3.7 which is copied to x64\Release and x64\Debug after build. So, you do not install a separate Python. If you need to install Python, don't change the system environment settings. Otherwise, Codiac service may load your installed Python instead of Codiac Python 3.7.

8.2 How to create Python AF extension function?

1. Open CodiacSDK.PythonWrapper\py_modules in VS Code.
2. Create a new Python script (Ctrl-N and save)
3. Create a class derived from AliasFrameworkModuleBase class and add imports to py_codiac_sdk_api and py_af_classes for helper functions.

```
ExampleModule.py > ExampleModulePY > Email
1   import json
2   from Modules.python.afmodulebase import AliasFrameworkModuleBase
3
4   import plugins.py_codiac_sdk_api as sdk_api
5   import plugins.py_af_classes as af_class
6
7   AF_MODULE = 'ExampleModulePY'
8
9   class ExampleModulePY(AliasFrameworkModuleBase):
10
```

4. Create methods that are going to be executed by AF. Below is an example of two empty. Functions. See examples in Helper functions on how to make tasks or get\set the values of aliases.

```
class ExampleModulePY(AliasFrameworkModuleBase):

    def Calculation1(self):
        # some calcualtions here
        pass

    def Task1(self):
        # some tasks here
        pass
```

5. Before the functions can exit, the alias of extension function must be updated. For Multi Generated alias, set the alias to vector of AliasValue objects. For Custom Generate alias, set the alias to an Alias object. Below is an example:

```
def Calculation1(self):
    # some calcualtions here

    output_aliases = af_class.std_vector_aliasvalue()
    av1 = af_class.AliasValue()
    av1.set_value_double(1.2345)
    output_aliases.append(av1)
    av2 = af_class.AliasValue()
    av2.set_value_double(3.1416)
    output_aliases.append(av2)

    self.insert_in_multi_map('Multi.ExampleModulePY.Calculation1', output_aliases)


def Task1(self):
    # some tasks here

    av = af_class.AliasValue()
    av.set_value_string("Completed a task")
    self.insert_in_map('Custom.ExampleModulePY.Task1', av)
```

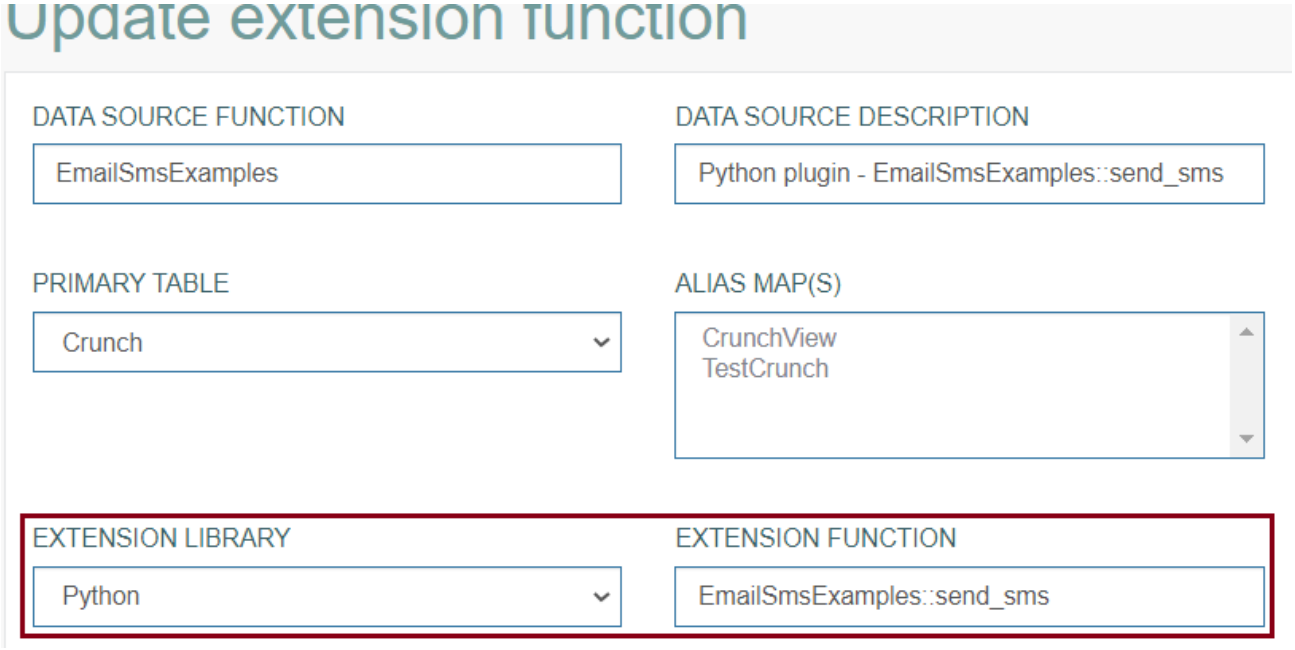
6. Copy the new Python scripts to *<service directory>*\Modules\python directory.

7. Finally, add the Custom\Multi Generated alias of your AF extension function in Codiac builder.
    a) Add your Custom\Multi Generated alias to Alias setup
    b) Configure your screen to include your Custom\Multi Generated alias
    c) Execute screen in Codiac render to test your AF extension function

9 Python extension functions (plugins)

The extension functions are configured in PHP client. Screenshot below is an example setup which specifies the class and the function to execute. This is different from C++ setup which specifies the user module name.

The CodiacSDK.PythonWrapper service handles the loading and execution of Python script. So, this service is called instead of user module.



Let say, we have “MyPlugin.py” script having a “Foo” class with “bar” method. To set it up as an extension function,

1. In PHP client, go to **System → Extension function → Create**
2. Select “Python” in **EXTENSION LIBRARY**
3. Put class name and function name in EXTENSION FUNCTION with double colon in between (Example: “Foo::bar”)

The setup will look like below.

Note that the script name (“MyPlugin.py”) is not use in the setup.

9.1 Required tools

- Visual Studio Code - https://code.visualstudio.com/
- Visual Studio Code: Python extension by Microsoft

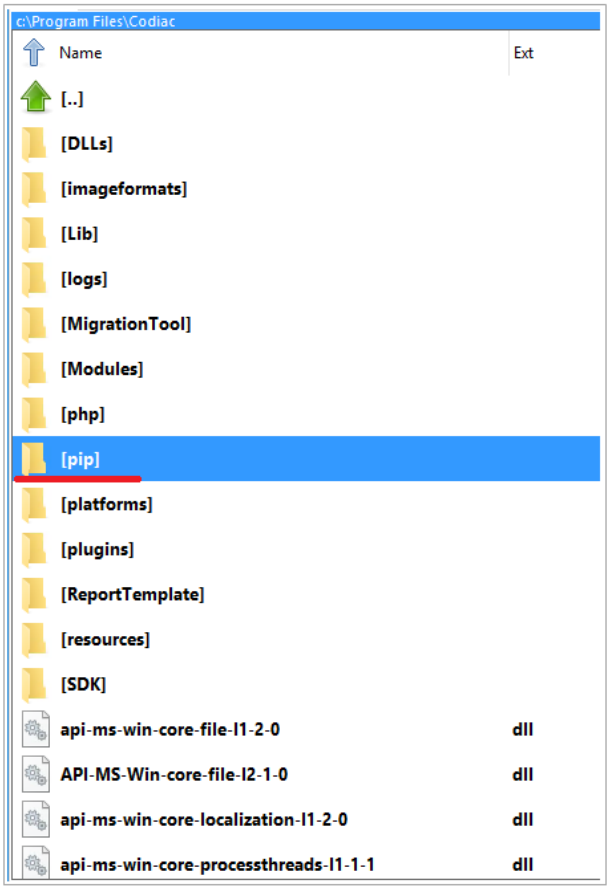
The Codiac.Service\~external directory contains Python 3.7 which is copied to x64\Release and x64\Debug after build. So, you do not install a separate Python. If you need to install Python, do not

change the system environment settings. Otherwise, Codiac service may load your installed Python instead of Codiac Python 3.7.

9.1.1 PIP package manager

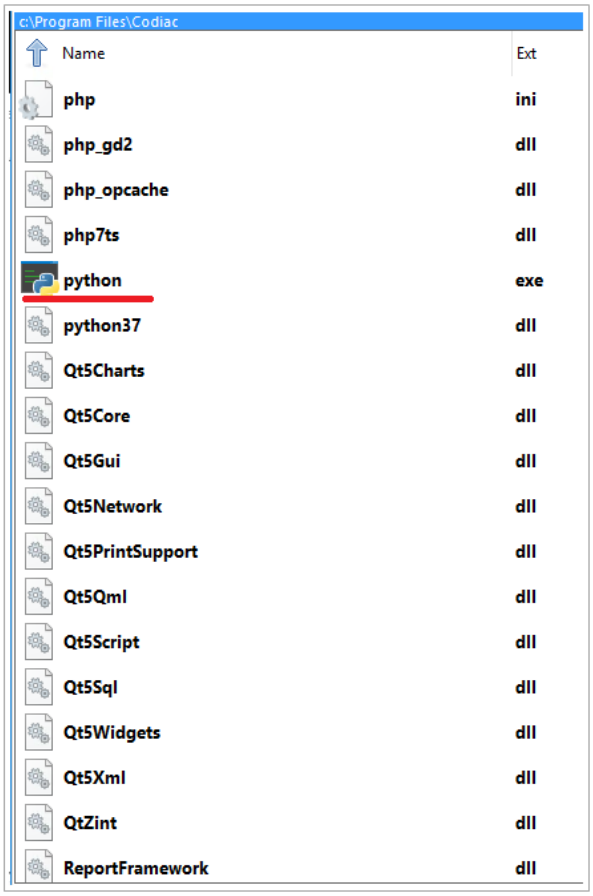
PIP (Preferred Installer Program) is the default package manager for Python, used for installing and managing software packages written in Python. PIP allows users to easily install, upgrade, and uninstall Python packages, as well as manage dependencies between packages.

The PIP folder is placed in the Codiac root folder.



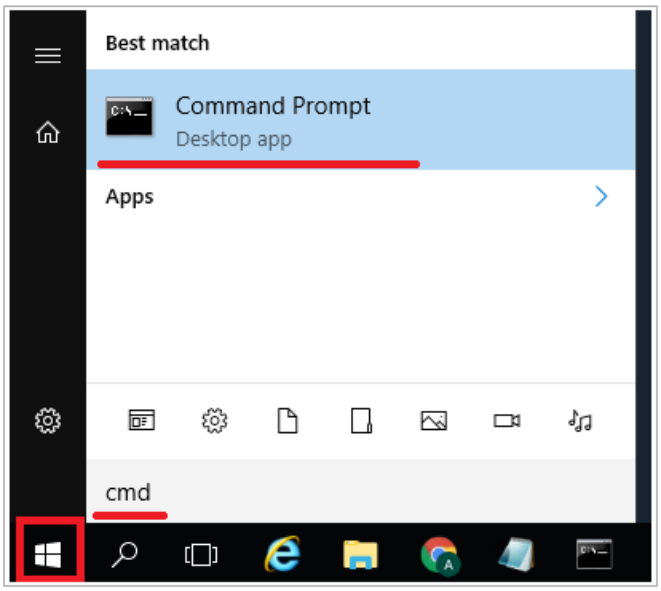
In the Codiac root folder, there is a built-in **python.exe** instance that is also configured specifically for the Codiac service.

Note that only this built-in **python.exe** instance should be used to guarantee compatibility between the installed package versions and the Codiac service.

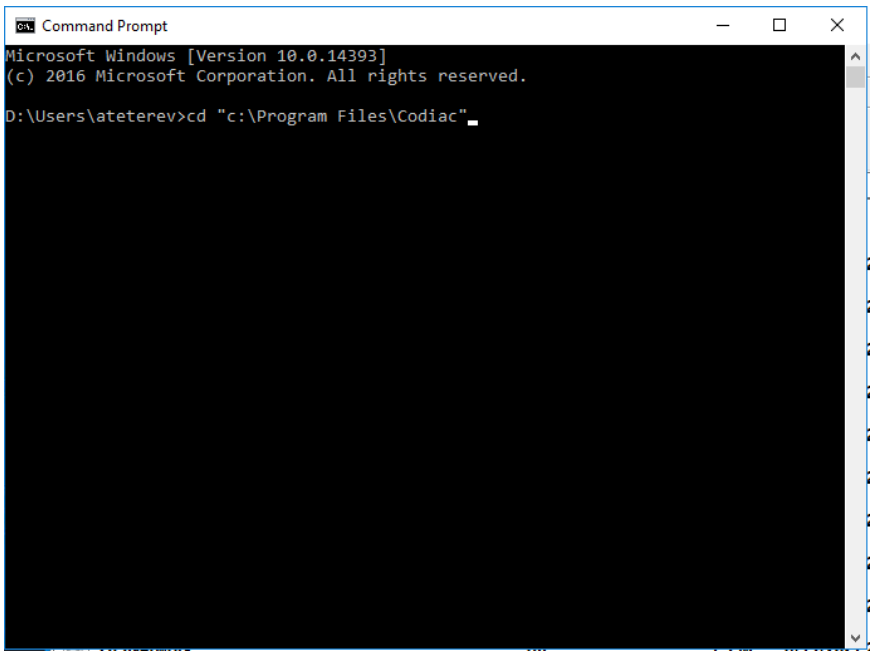


To use PIP in the Codiac service, perform the following steps:

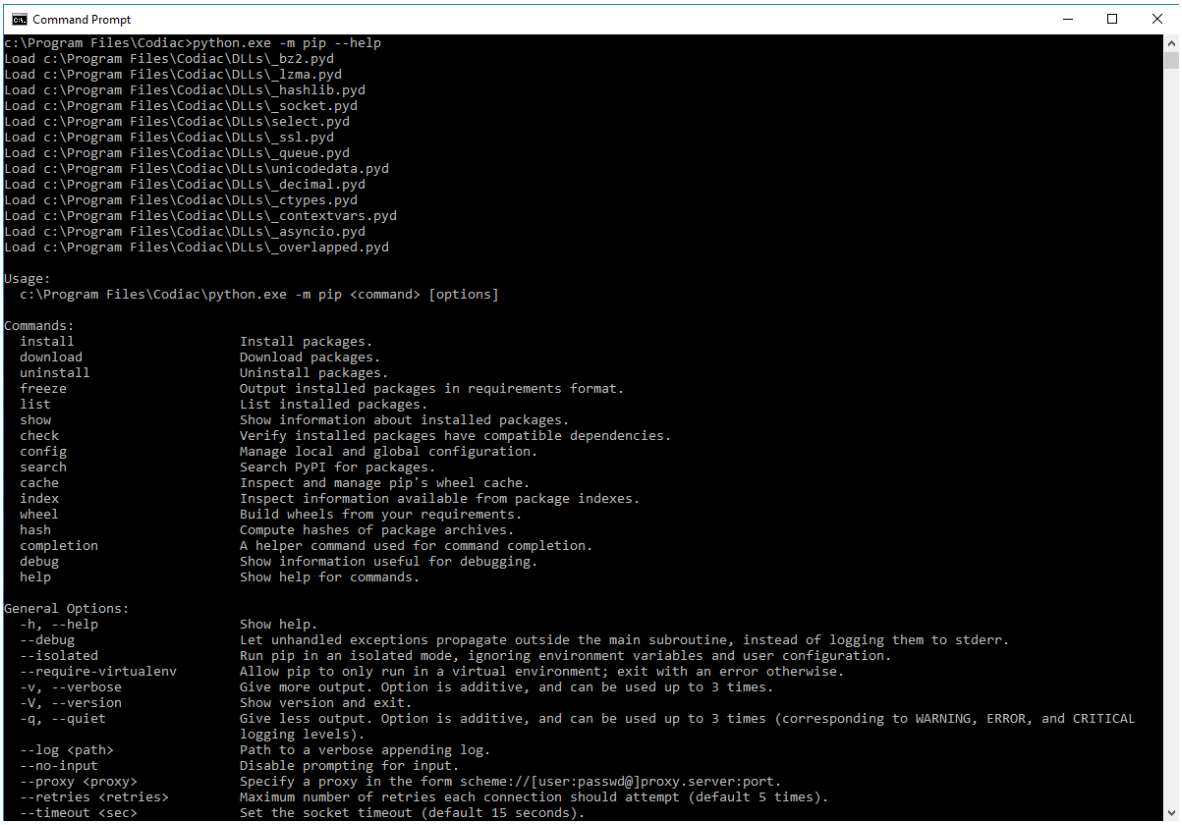
1. Launch the Command Prompt utility.



2. Change the working directory to the **Codiac** service root folder.



3. Use **Help** to see a list of tips that relate to PIP.
   Type the **python.exe -m pip –help** command and press the **Enter** key. The list will be shown.



In the opened list, users can see usage tips, commands, and general options examples.
For example, if it is necessary

- to install **pyparsing** package, type the following command in the Command Prompt:
  python.exe -m pip install pyparsing
- to get the installed packages list, type the following command in the Command Prompt:
  python.exe -m pip list

- to uninstall **pyparsing** package, type the following command in the Command Prompt:
  python.exe -m pip uninstall pyparsing

More details about supported commands can be found at the following link: https://pip.pypa.io/en/stable/cli/pip/.

9.2 Structure of Python plugins

The Python plugins' structure follows the Codiac service structure. Meaning that it has "configure" function, "sdk_module_info" function, and "func_controller" function. For convenience, these functions are defined in BasePlugin class so that the author of Python plugin doesn't need to define these functions. So, Python plugins just need to derived from BasePlugin class.

9.2.1 The plugin class

The Python plugin class derives from BasePlugin class and contains the implementations for plugin functions. Furthermore, the constructor defines the list of extension functions with reference to Python methods.

```
class AliasFrameworkExamples(BasePlugin):

    def __init__(self):
        super().__init__(MODULE_NAME)
        self.plugin_functions = {
            'get_crunch_maturity_date': {
                'func': self.get_crunch_maturity_date,
                'func_descr': 'plugin AF example that evaluates single value aliases'
            },
            'get_crunchf_data': {
                'func': self.get_crunchf_data,
                'func_descr': 'example AF plugin that evaluates multi values aliases'
            },
```

For example (image below), the constructor adds "get_crunch_maturity_date" and "get_crunchf_data" (with reference to methods) to the plugin_functions dictionary. This dictionary is defined in BasePlugin class. So, when func_controller function is called with the extension function name (for example: "get_crunch_maturity_date"), the func_controller function looks for the extension function name in plugin_functions dictionary and then calls the referenced method.

9.3 How to create a Python plugin?

1. Open CodiacSDK.PythonWrapper\py_scripts in VS Code.
2. Create a new Python script (Ctrl-N and save).
3. Create a class derived from BasePlugin class and base class constructor with plugin name.

```python
from base_plugin import BasePlugin

MODULE_NAME = "ExamplePluginPY"

class ExamplePluginPY(BasePlugin):

    def __init__(self):
        super().__init__(MODULE_NAME)

```

4. Create methods for the plugin class. Below are examples of empty plugin functions. See the example of Python helper functions on how to add implementations to plugin functions.

```python
class ExamplePluginPY(BasePlugin):

    def __init__(self):
        super().__init__(MODULE_NAME)

    def my_first_plugin(self, json_param):
        # some task here
        pass

    def another_plugin(self, json_param):
        # some task here
        pass
```

5. Add the new methods to plugin_functions dictionary. This means to add the plugin function name and the reference to the method. Don't declare the plugin_function dictionary in plugin class. This map is already declared in BasePlugin class.

   Follow the structure below.

```python
def __init__(self):
    super().__init__(MODULE_NAME)

    self.plugin_functions = {
        'my_first_plugin': {
            'func': self.my_first_plugin,
            'func_descr': 'my first plugin'
        },
        'another_plugin': {
            'func': self.another_plugin,
            'func_descr': 'another plugin'
        }
    }
```

6. Copy the new Python scripts to *<service directory>*\plugins\python directory.
7. Finally, add your plugin function in Codiac builder.
   a) Add your plugin functions to extension function setup.

b) Configure your screen to include the extension function.

c) Execute screen in Codiac render to test your plugin.

10 Python helper functions

These Python functions will allow you to access to Alias Framework, query database, send email, send SMS message, and execute Codiac services.

Note that some functions are only available for AF extension functions, some are only for plugins and other functions are for both.

10.1 AF helper functions for AF extension functions

10.1.1 get_from_map(*alias_name*)

- **Purpose:** return AliasValue object of an alias
- **Example:** This example returns the AliasValue object of "Alias.EmailSendTests.language". With the AliasValue object, the string value can be retrieve using value_to_string method.

```
avLanguage = self.get_from_map("Alias.EmailSendTests.language")
language = avLanguage.value_to_string()
if language == "PY":
```

10.1.2 get_from_multi_map(alias_name)

- **Purpose:** return array of AliasValue objects of multi-value alias
- **Example:** This example returns the array of AliasValue objects of "Array.CrunchF.Amt".Then, iterate on that list of AliasValue objects for processing each item.

```
amounts = self.get_from_multi_map('Array.CrunchF.Amt')
for amount in amounts:
    value = amount.value_to_string()
```

10.1.3 insert_in_map(key, value)

10.1.4 replace_or_insert_in_map(key, value)

- **Purpose:** insert alias (with AliasValue object) in evaluated aliases
- Example:This example creates an AliasValue object and add it in evaluated aliases.

```
av = af_class.AliasValue()
av.set_value_string(value)
self.insert_in_map('Custom.ExampleModulePY.Email', av)
```

- Note: The insert_in_map is similar to replace_or_insert_in_map except that it doesn't insert if the Alias name exist in evaluated aliases.

10.1.5 insert_multi_map(key, value)

10.1.6 replace_or_insert_multi_map(key, value)

- **Purpose:** insert multi-value Alias (with vector of AliasValue objects) in evaluated aliases which
- **Example:** This example creates list of AliasValue objects and add it in evaluated aliases.

```
output_aliases = af_class.std_vector_aliasvalue()
fetched_sql = sql_result.fetch_one()
for item in fetched_sql:
    av = af_class.AliasValue()
    av.set_value_string(item)
    output_aliases.append(av)

self.insert_in_multi_map('Multi.ExampleModulePY.SqlSelectOtherDb', output_aliases)
```

- **Note:** The insert_in_map is similar to replace_or_insert_in_map except that it doesn't insert if the Alias name exist in evaluated aliases.

10.2 Alias Value class

10.2.1 plugins.py_af_classes.AliasValue constructor

- **Purpose:** create AliasValue object

10.2.2 set_value_string(value), set_value_int(value), set_value_double(value)

- **Purpose:** set AliasValue object with double value, integer, or string

10.2.3 value_to_int(), value_to_double(), value_to_string()

- **Purpose:** convert the Alias value object to integer, to double or return the alias string.

10.2.4 is_value_null()

- **Purpose:** check if the Alias value is uninitialized

10.2.5 mark_for_deletion()

- **Purpose:** delete database table row pointed by Alias value after execution of AF extension function

10.2.6 is_mark_for_deletion()

- **Purpose:** check if the Alias value is for deletion

Note: See the helper functions for AF for example usage of AliasValue class.

10.3 AliasView class for Python plugins

10.3.1 py_codiac_sdk_api.AliasView(aliasView, aliasInput, ruleInput,

userInformation, dataPoolLoader) constructor

- **Purpose:** create AliasView instance that evaluates alias view

10.3.2 Example

- This example creates and AliasView instance and evaluates “CrunchView” alias view with input “Alias.Crunch.Contract” alias having value of “SAMPLEVUL”. The evaluate method stores the evaluated data to the evaluated_data['returnData'] argument.

```
# name of alias view is CrunchView and input Contract is SAMPLEVUL
alias_views = json.dumps([ 'CrunchView' ])
alias_input = json.dumps({ 'Alias.Crunch.Contract': 'SAMPLEVUL' })

# create AliasView object
av = AliasView(alias_views, alias_input, func_param, func_param, self.pool_loader)

# evaluate CrunchView alias view (alias framework)
evaluated_data = json.loads(av.evaluate())

status = evaluated_data['result']
out_data = evaluated_data['returnData']
```

10.4 JSON request helper functions for Python plugins

These helper functions parse the JSON request and provide convenient way to retrieve and update the values of aliases. This is are normally use for plugins that going to be assigned as pre-extension functions.

10.4.1 py_codiac_sdk_api.plugin_helpers.get_alias_value(alias)

- **Purpose:** retrieve the string value of an alias
- **Example:** This example retrieve the value of “Alias.Crunch.Contract” alias from JSON request.

```
contract = plugin_helpers.get_alias_value('Alias.Crunch.Contract')
```

10.4.2 py_codiac_sdk_api.plugin_helpers.set_alias_value()

- **Purpose:** modify the value of an alias
- **Example:** This example modifies “Alias.Crunch.Notes” alias of the JSON request to “Updated by plugin”.

```
plugin_helpers.set_alias_value('Alias.Crunch.Notes', 'Updated by plugin')
```

10.4.3 py_codiac_sdk_api.plugin_helpers.get_alias_array_size()

- **Purpose:** returns the size of alias array

- Note: See the example of get_alias_array_value function for usage of get_alias_array_size function

10.4.4 py_codiac_sdk_api.plugin_helpers.get_alias_array_value(index, alias)

- **Purpose:** retrieve the string value of alias from an array index
- **Example:** This example retrieves the value of each "Array.CrunchF.Amt" alias from JSON request.

```
# get the array size of modified data from patch_json
num_rows_for_update = plugin_helpers.get_alias_array_size()
for i in range(num_rows_for_update):
    # get Alias values by array index
    amount = plugin_helpers.get_alias_array_value(i, 'Array.CrunchF.Amt')
```

10.4.5 py_codiac_sdk_api.plugin_helpers.set_alias_array_value(index, alias, value)

- **Purpose:** modify the value of an alias from an array index
- **Example:** This example replaces the value of "Array.CrunchF.Amt" with the value of amount_number.

```
plugin_helpers.set_alias_array_value(i, 'Array.CrunchF.Amt', str(amount_number))
```

10.4.6 py_codiac_sdk_api.plugin_helpers.get_aliases()

- **Purpose:** return aliases (excludes alias array) from values of aliases of JSON request (in list of AliasValue objects in key\data objects)
- **Example:** This example iterate on list of aliases and prints the each values.

```
# Get map of aliases. The key is the alias name and value is its alias value.
aliases = plugin_helpers.get_aliases()
for alias in aliases:
    alias_name = alias.key()
    alias_value = alias.data()
    print(f'Alias: {alias_name} Value: {alias_value.value_to_string()}')
```

10.4.7 py_codiac_sdk_api.plugin_helpers.get_alias_array()

- **Purpose:** return entire alias array of JSON request (in list(key\data) of list of AliasValue objects)
- **Example:** This example iterate on map of aliases then iterate on vector of AliasValue objects. Then, print the value of aliases. The get_alias_array() function is another way to get the values of alias array in addition to get_alias_array_value() function.

```
# Get map of alias arrays. The key is the alias array name and value is its list of values.
alias_array = plugin_helpers.get_alias_array()
for alias in alias_array:
    alias_name = alias.key()
    print(f'Alias: {alias_name}')

    alias_values = alias.data()
    for alias_value in alias_values:
        print(f' has {alias_value.value_to_string()}')
```

10.5 Message helper functions for Python plugins

10.5.1 py_codiac_sdk_api.plugin_helpers.is_confirmed(message_code)

- **Purpose:** Check if the JSON request has confirmed message_code. This helper function is use with plugin_helpers.INFO and plugin_helpers.WARNING.
- Note: See the examples of plugin_helpers.INFO and plugin_helpers.WARNING for usage of IsConfirmed function.

10.5.2 py_codiac_sdk_api.plugin_helpers.INFO(message_code, message)

- **Purpose:** JSON response with INFO message
- **Purpose:** This example sends back a message response to client with an id of 1.

```
if not plugin_helpers.is_confirmed(1):
    return plugin_helpers.INFO(1, 'Starting a task')
```

- **Client output:**

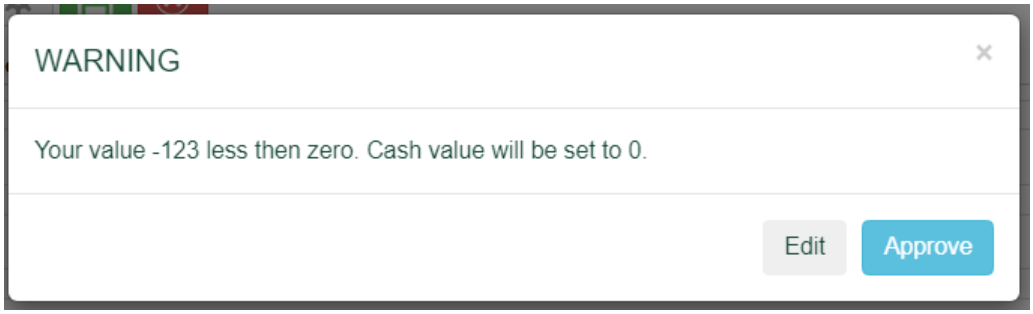
- Note: The client will resend the request after it has acknowledge the Info message. So, the plugin function must check the message_code to skip the INFO message.

10.5.3 py_codiac_sdk_api.plugin_helpers.WARNING(message_code, message)

- **Purpose:** JSON response with WARNING message
- **Example:** This example sends back a warning message response to client with an id of 2.

```
if plugin_helpers.is_confirmed(2):
    cash = 0.0
else:
    message = f'Your value {cashValue} less then zero. Cash value will be set to 0.'
    return plugin_helpers.WARNING(2, message)
```

- **Client output:**



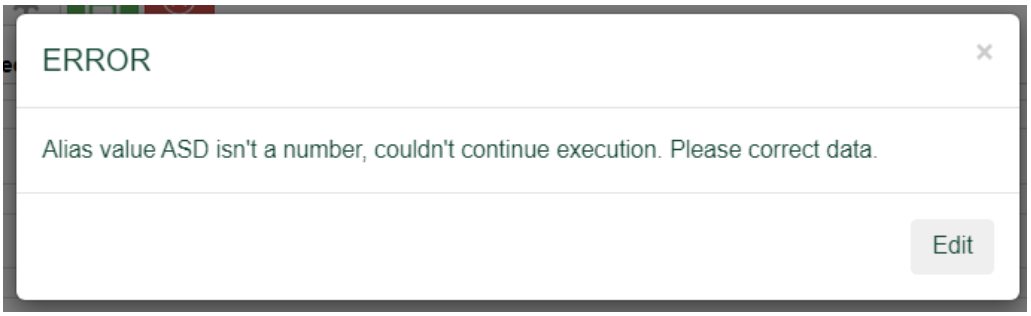
- Note: The client will resend the request after it has acknowledge the Warning message. So, the plugin function must check the message_code to skip the WARNING message.

10.5.4 py_codiac_sdk_api.plugin_helpers.ERROR(message_code, message)

- **Purpose:** JSON response with ERROR message
- **Example:** This example sends back an error message response to client.

```
message = f'Alias value {cashValue} isn''t a number, couldn''t continue execution. Please correct data.'
return plugin_helpers.ERROR(4, message)
```

- **Client output:**



- Note: The client doesn't resend the request after it has acknowledge the Error message.

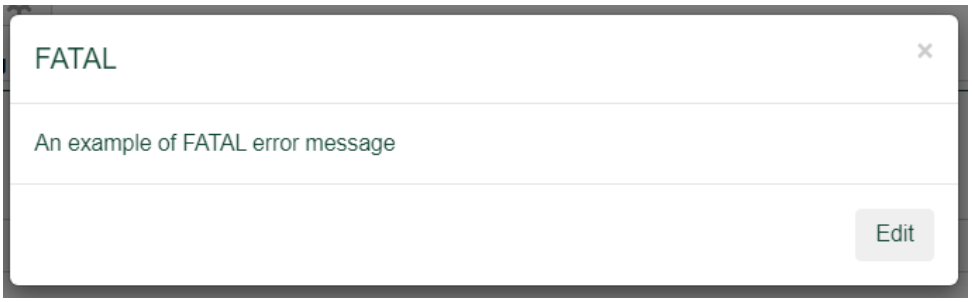
10.5.5 py_codiac_sdk_api.plugin_helpers.FATALmessage_code, message)

- **Purpose:** JSON response with FATAL message

- **Example:** This example sends back an error message response to client.

```
return plugin_helpers.FATAL(8, 'An example of FATAL error message')
```

- **Client output:**



- **Note:** The client doesn't resend the request after it has acknowledge the Error message.

10.6 Document management helper functions for Python plugins

10.6.1 sdk_api.copy_document_to_device(filePath, fileDescription, documentFamily, documentCategory)

- **Purpose:** Copies document to a specific device, the family and category are optional parameters, if they are not specified the document is copied according to the parameters in the JsonKeeper.
- **Return:** Returns the file container ID if successful, otherwise an empty string is returned.
- **Note:** The example for this function is provided together with the "add_document_to_return_list" function example.

10.6.2 sdk_api.add_document_to_return_list(documentID)

- **Purpose:** Adds the file container ID to the JsonKeeper and later in the response of the extension function.
- **Return:** Returns "true" if successfully executed, otherwise "false" is returned.
- **Example:**

```
isAdded = False
fileID = ''
filePath = 'C:\\DocumentManagementTest.txt'
fileDescription = 'description'

documentFamily = 'test'
documentCategory = 'test'

# copies the document to the device specified on the screen
fileID = sdk_api.copy_document_to_device(filePath, fileDescription)
# adds the file container ID in the response
isAdded = sdk_api.add_document_to_return_list(fileID)
print('File container ID: ' + fileID + '. Added to return list: ' + str(isAdded) + '.')

# copies the document to the device specified in the parameters
fileID = sdk_api.copy_document_to_device(filePath, fileDescription, documentFamily, documentCategory)
# adds the file container ID in the response
isAdded = sdk_api.add_document_to_return_list(fileID)
print('File container ID: ' + fileID + '. Added to return list: ' + str(isAdded) + '.')

return json.dumps(BasePlugin.get_success_response('OK'))
```

10.6.3 sdk_api.get_document_from_device(fileContainerID, filePath)

- **Purpose:** Loads the document with the specified file container ID in the specified path.
- **Return:** Returns “true” if successfully executed, otherwise “false” is returned.
- **Example:**

```
isLoaded = False
fileID = '1111'
filePath = 'D:\\Users\\spoposki\\Desktop\\testFile.jpg'

# loads the document with the specified file container ID in the specified path
isLoaded = sdk_api.get_document_from_device(fileID, filePath)
print('File container ID: ' + fileID + '. Is loaded: ' + str(isLoaded) + '. Location: ' + filePath)

return json.dumps(BasePlugin.get_success_response('OK'))
```

10.6.4 sdk_api.delete_document_from_device(fileContainerID, performHardDelete)

- **Purpose:** Sets the file container “Active” flag to “false” or deletes the file container completely, it depends on the value of the “performHardDelete” parameter.
- **Returns:** Returns “true” if successfully executed, otherwise “false” is returned.
- **Example:**

```
isDeleted = False
fileID = '1111'

# performs a soft delete on the specified file container
isDeleted = sdk_api.delete_document_from_device(fileID, False)
print('File container ID: ' + fileID + '. Is soft deleted: ' + str(isDeleted))

# performs a complete delete on the specified file container
isDeleted = sdk_api.delete_document_from_device(fileID, True)
print('File container ID: ' + fileID + '. Is completely deleted: ' + str(isDeleted))

return json.dumps(BasePlugin.get_success_response('OK'))
```

10.6.5 sdk_api.search_for_document(documentFamily, documentCategory, kp1, kp2, kp3, kp4, kp5, description)

- **Purpose:** Searches for file containers using the function parameters as search parameters. If a function parameter is an empty string that field will not be included in the search.
- **Return:** Returns serialized JSON array containing information about the file containers that match the search parameters.
- **Example:**

```
fileRootPath = 'D:\\Users\\spoposki\\Desktop\\'
description = 'test1'

# searches for the documents that have the value "test" in description
files = sdk_api.search_for_document('', '', '', '', '', '', '', description)
print('Files: ' + files)
# parses the response and iterates through the array
filesArray = json.loads(files)
for file in filesArray:
    isLoaded = False
    filePath = fileRootPath + file['container_file_name']
    # loads the document to disk
    isLoaded = sdk_api.get_document_from_device(str(file['id']), filePath)
    print('File container ID: ' + str(file['id']) + '. Is loaded: ' + str(isLoaded) + '. Location: ' + filePath)

return json.dumps(BasePlugin.get_success_response('OK'))
```

10.6.6 sdk_api.set_document_key_parts(fileContainerID, kp1, kp2, kp3, kp4, kp5)

- **Purpose:** Sets the file container key parts.
- **Return:** Returns “true” if successfully executed, otherwise “false” is returned.
- **Example:**

```
isSet = False

fileID = '1111'
kp1 = 'test_kp1'
kp2 = 'test_kp2'
kp3 = 'test_kp3'
kp4 = 'test_kp4'
kp5 = 'test_kp5'

# sets the key part
isSet = sdk_api.set_document_key_parts(fileID, kp1, kp2, kp3, kp4, kp5)
print('Key parts set: ' + str(isSet))

return json.dumps(BasePlugin.get_success_response('OK'))
```

10.7 Database helper functions for Python plugins and PythonAF extension functions

10.7.1 py_codiac_sdk_api.plugin_helpers.DBCursor

- **Purpose:** store the query results of execute_sql; and able to fetch row sequentially from that query results
- **Methods:**
  - size() returns number of rows
  - set_position(position) set the position of cursor in query results
  - get_position() return the position of cursor in query results
  - fetch_one() returns row (in list of columns); and next fetch_one() call returns next row
  - get_column_names() returns the column names of query results. The column names are in alphabetic order.
- **Note:** See the example of execute_sql function for usage

10.7.2 py_codiac_sdk_api.plugin_helpers.execute_sql(sql_query, db_connection)

- **Purpose:** query the database by running SQL query to Codiac database or to other database (with DB connection information)
- **Note:** db_connection parameter is optional and default to empty which means to connect to Codiac database
- **Example:** This example run SQL query to Codiac database and store the first column to vector of AliasValue objects.

```
# execute SQL query
sql_result = sdk_api.execute_sql('SELECT "Crunch"."Contract" FROM "Crunch"')

# put results in evaluated alias
# note: std_vector_aliasvalue is use to store result aliases instead of python list
output_aliases = af_class.std_vector_aliasvalue()
for i in range(sql_result.size()):

    # get one row
    fetched_sql = sql_result.fetch_one()

    av = af_class.AliasValue()
    av.set_value_string(fetched_sql[0])
    output_aliases.append(av)
```

- **Example:** This example run SQL query to postgres database (with DB connection information) and store the first row to vector of AliasValue objects.

```
db_connector = sdk_api.DBConnector()
db_connector.set_db_type("Postgres")
db_connector.set_db_name("postgres")
db_connector.set_db_user("postgres")
db_connector.set_db_pass("[redacted]")
db_connector.set_db_port("5432")
db_connector.set_db_address("[redacted]")

# execute SQL query with database connection string
sql_result = sdk_api.execute_sql('SELECT * FROM "persons"', db_connector)

# put results in evaluated alias
# note: std_vector_aliasvalue is use to store result aliases instead of python list
output_aliases = af_class.std_vector_aliasvalue()
fetched_sql = sql_result.fetch_one()
for item in fetched_sql:
    av = af_class.AliasValue()
    av.set_value_string(item)
    output_aliases.append(av)
```

10.8 Other CodiacSDK helper functions for Python plugins and AF extension functions

10.8.1 py_codiac_sdk_api.plugin_helpers.send_email(to, subject, body)

- **Purpose:** send an email to recipient with subject and message
- **Example:** This AF extension function example sends an email with recipient from “Alias.EmailSendTests.recipient” alias, subject from “Alias.EmailSendTests.subject” alias and message from “Alias.EmailSendTests.body” alias.

```
avRecipient = self.get_from_map("Alias.EmailSendTests.recipient")
recipient = avRecipient.value_to_string()
avSubject = self.get_from_map("Alias.EmailSendTests.subject")
subject = avSubject.value_to_string()
avBody = self.get_from_map("Alias.EmailSendTests.body")
body = avBody.value_to_string()
sdk_api.send_email(recipient, subject, body)
```

10.8.2 py_codiac_sdk_api.plugin_helpers.send_email(to, subject, body, attachments)

- **Purpose:** Sends an email to recipient with subject, message and attachments. The attachment parameter is a delimited string of file container IDs.
- **Return:** A serialized JSON with details.
- **Example:**

```
to = 'slavkopoposki@gmail.com'
subject = 'SUBJECT'
body = 'BODY'

files = '1178;1179;1180'

# sends an email with attachments, the file that correspond to the id will the attached to the email
sdk_api.send_email(to, subject, body, files)

return json.dumps(BasePlugin.get_success_response('OK'))
```

10.8.3 py_codiac_sdk_api.plugin_helpers.send_sms(to, body)

- **Purpose:** send SMS message to recipient with message
- **Example:** This example send SMS message to 5556789090 (not a real phone number). Only US numbers are accepted.

```
sdk_api.send_sms('5556789090','This is a test message.')
```

10.8.4 py_codiac_sdk_api.plugin_helpers.generate_report(report_name, inline_json, is_save_to_xml)

- **Purpose:** generate report PDF file from report template (client report builder). Output PDF is saved in document repository.
- **Note:** The is_save_to_xml default to false; if set to true, the report are save to XML file and is not generated to PDF.
- **Example:** This example generate report from alias view data. The alias view data contains alias values in JSON format.

```
alias_view_data = json_params['alias_view_data']
inline_json = None
if type(alias_view_data) is list and len(alias_view_data) > 0:
    inline_json = json.dumps(alias_view_data[0])
else:
    inline_json = json.dumps(alias_view_data)

report_name = "Test CrunchF Report"
return self.generate_report(json_params, report_name, inline_json)
```

- **Example:** Another example that generate report from report template “Test CrunchF Report” with values from dependent multi-value Aliases. Note that the function has to create a JSON input string from aliases.

```
# dependency aliases
contracts = self.get_from_multi_map('Array.CrunchF.Contract')
descriptions = self.get_from_multi_map('Array.CrunchF.Description')
fundNos = self.get_from_multi_map('Array.CrunchF.FundNo')
amounts = self.get_from_multi_map('Array.CrunchF.Amt')

# list of array aliases in json
json_input = {}

json_input['Array.CrunchF'] = []
for i in range(len(contracts)):
    item = {
        'Array.CrunchF.Contract': contracts[i].valueToString(),
        'Array.CrunchF.Description': descriptions[i].valueToString(),
        'Array.CrunchF.FundNo': fundNos[i].valueToString(),
        'Array.CrunchF.Amt': amounts[i].valueToString()
    }
    json_input['Array.CrunchF'].append(item)

# generate report
sdk_api.generate_report('Test CrunchF Report', json.dumps(json_input))
```

10.8.5 py_codiac_sdk_api.plugin_helpers.generate_report(report_name, inline_data, delimiter, is_save_to_xml)

- **Purpose:** generate report PDF file from report template (client report builder)
- **Note:** The is_save_to_xml default to false; if set to true, the report are save to XML file and is not generated to PDF.
- **Example:** This example generate report from report template “Test CrunchF Report” with values from vector of delimited values. The delimiter used '|'.

```
report_name = 'Test CrunchF Report - inline data'
inline_data = [
    'Contract|FundNo|Description|Amount',
    'Contract A|CA1|Test contract A - 1|123.45',
    'Contract A|CA2|Test contract A - 2|3.45',
    'Contract B|CB1|Test contract B - 1|645',
    'Contract C|CC1|Test contract C - 1|7.89'
]

return self.generate_report(json_params, report_name, inline_data, delimiter='|')
```

10.8.6 py_codiac_sdk_api.plugin_helpers.execute_json(jsonRequest)

- **Purpose:** call Codiac service by executing JSON request
- **Example:** This example calls JobsScheduler::GetJobsScheduleList to get list of jobs. A JSON request is created with func_name parameter and lib_name parameter.
- **Note:** The user\session parameters are filled up at service process.

```
# create json request
json_request = {
    'requestbody' : {
        'func_name' : 'GetJobScheduleList',
        'lib_name' : 'CodiacSDK.zJobsScheduler',
    }
}

# execute service to get Jobs
json_response = sdk_api.execute_json(json.dumps(json_request))
```

10.9 SQL query with Python native database modules

Codiac SDK provides database helper functions that can access Codiac database and other databases with connection string. In addition, the user can directly import and call the python module to query the database.

The DirectSqlExamples plugins and ExampleModule AF extension functions contain example source code on how to query the databases with native database modules

- **PostgreSQL**

  The example source codes uses psycopg2 module to query PostgreSQL database. Install the psycopg2 module so that module will be available in the Python script. To install:

  ```
  pip install psycopg2
  ```

  Below is the sequence for psycopg2 to query database:

  1. Call “connect” function with database connection information to connect to database
  2. Call “cursor” function on connection object to create cursor object

3. Call “execute” function with SQL query string on cursor object
4. Call “commit” function to execute the query
5. Call “fetchall” function to retrieve the query results
6. The fetched results are index by column number.
7. Call “close” function on cursor object to close the cursor
8. Call “close” function on connection object to close the database connection

See the PostgreSQL function of direct_sql_examples.py and PostgeSQL function of ExampleModule.py for full example.

See the https://www.psycopg.org/ for full details on how to use psycopg module.

- **MySQL**

  The example source codes uses mysql.connector module to query MySQL database. Install the mysql.connector module so that module will be available in the Python script. To install:

  ```
  pip install mysql-connector-python
  ```

  Below is the sequence for mysql.connector to query database:

  1. Call “connect” function with database connection information to connect to database
  2. Call “cursor” function on connect object to create cursor object
  3. Call “execute” function with SQL query string on cursor object to execute the query
  4. Iterate on cursor to fetch the query result row
  5. Close the cursor object and connect object

  See the MYSQL function of direct_sql_examples.py and MySQL function of ExampleModule.py for full example.

  See the https://dev.mysql.com/doc/connector-python/en/ for full details on how to use mysql.connector module.

- **MS SQL Server**

  The example source codes uses pymssql module to query MS SQL Server database. Install the pymssql module so that module will be available in the Python script. To install:

  ```
  pip install pymssql
  ```

  Below is the sequence for pymssql to query database:

  1. Call “connect” function with database connection information to connect to database
  2. Call “cursor” function on connect object to create cursor object
  3. Call “execute” function with SQL query string on cursor object to execute the query
  4. Call “fetchone” function fetch the query result row
  5. Call “commit” function on connection object to confirm any changes on the database (**Example:** create, update, insert, drop).
  6. Call “close” function to close the connection to database.

See the MSSQL function of direct_sql_examples.py and MsSQLServer function of ExampleModule.py for full example.

See the https://github.com/pymssql/pymssql for full details on how to use pymssql module.

- **Oracle SQL**

  The example source codes uses cx_Oracle module to query Oracle SQL database. Install the cx_Oracle module so that module will be available in the Python script. To install:

  ```
  pip install cx_Oracle
  ```

  Below is the sequence for mysql.connector to query database:

  1. Call “connect” function with database connection information to connect to database
  2. Call “cursor” function on connect object to create cursor object
  3. Call “execute” function with SQL query string on cursor object to execute the query
  4. Iterate on cursor to fetch the query result row
  5. Call “commit” function on connection object to confirm any changes on the database (**Example:** create, update, insert, drop).

  See the OracleSQL function of direct_sql_examples.py and OracleSQL function of ExampleModule.py for full example.

  See the https://oracle.github.io/python-cx_Oracle/ for full details on how to use cx_Oracle module.

11 Python Debugging with Visual Studio Code

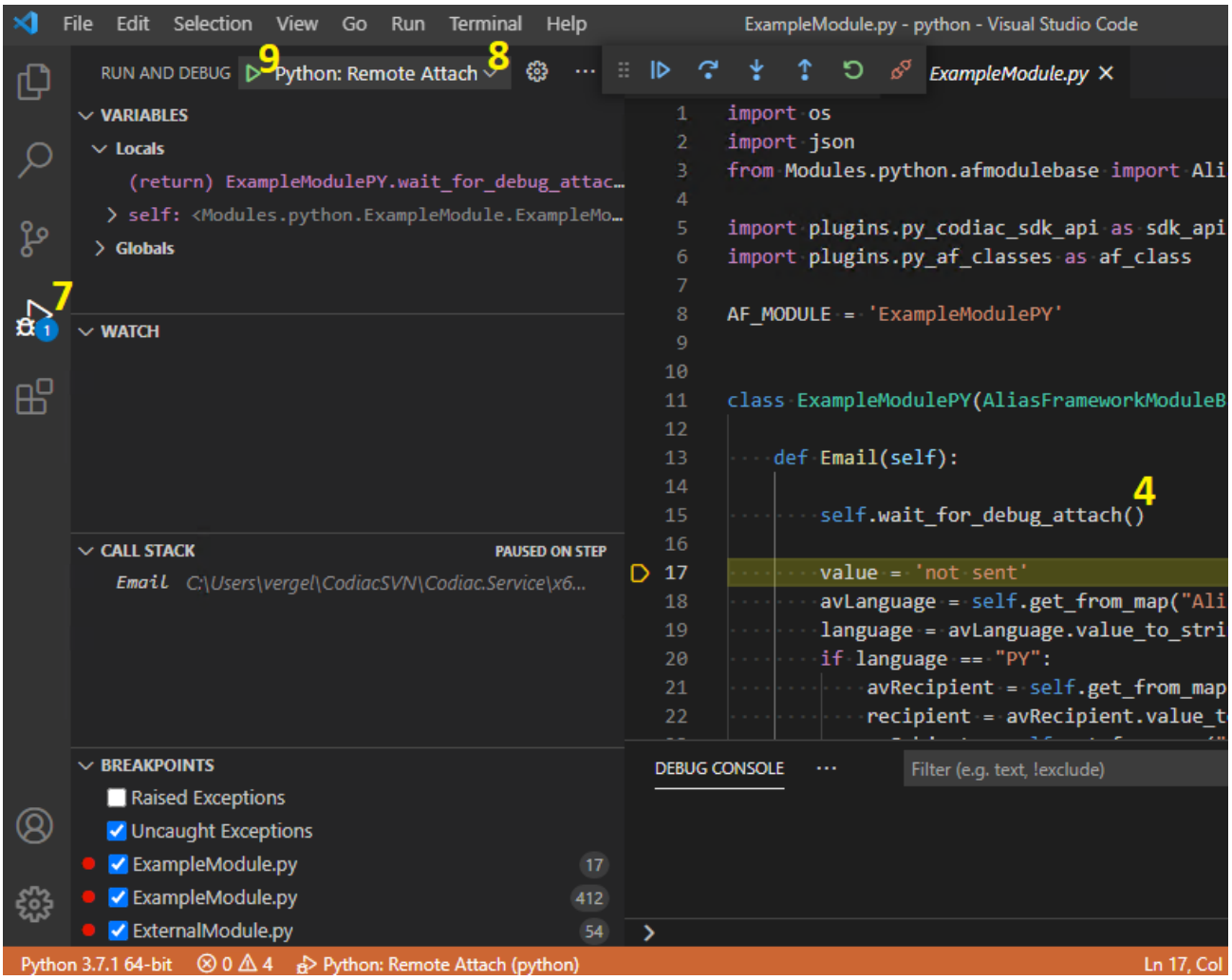
Before starting the debugger, create launch.json file for Python debugging.

```
{
  // Use IntelliSense to learn about possible attributes.
  // Hover to view descriptions of existing attributes.
  // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Python: Remote Attach",
      "type": "python",
      "request": "attach",
      "connect": {
        "host": "127.0.0.1",
        "port": 3000
      },
      "pathMappings": [
        {
          "localRoot": "${workspaceFolder}",
          "remoteRoot": "."
        }
      ],
      "justMyCode": false
    }
  ]
}
```

While the service is running, starts VS Code debugger with AF extension module:

1. Run the VS Code in administrator account
2. Open the Python directory to be debugged
   - AF extension function: x64\<*Debug*|*Release*>\Modules\python
   - plugins: x64\<*Debug*|*Release*>\plugins\python
3. Open the Python script to be debugged
4. Add breakpoint by inserting the “self.wait_for_debug_attach()” and save the file.
5. Execute the Python plugin or Python AF extension function.
6. Wait for a few seconds or wait for console to print “waiting for debugger”.
7. Click the debug icon on left side bar
8. Select the Python: Remote Attach

9. Click the green play button (or F5) to start debugging
10. At this point, VS Code is attached to the debugger, shows the current being executed and bottom part turns orange.



11. You can step through the codes by pressing the F10 (step next), F11 (step into), and F5 (continue) keys or disconnect (orange disconnect icon). You can add watches or see local watches.

12 Remote Debugging Python plugin and AF extension functions

In the Python language, remote debugging is almost identical to local debugging.
In the installed the **Python Visual Studio Debugger** engine (ptvsd), updated the following code line:

ptvsd.enable_attach(address=('localhost', 3000))

Where the **localhost** and local **port** values should be changed to the remote machine IP address and port.
Then, perform the debugging steps as for the local debugging described above in the Debugging section.

13 PHP Alias Framework extension module

The Alias Framework (AF) service can evaluate custom generated aliases or multi generated aliases. This is done by going through the alias name structure:

AliasType.ModuleName.FunctionName

Depending on the AliasModuleType, the Custom\Multi Generated alias can be C++ function, Python function, JavaScript function, or PHP function.

For example, the “Custom.ExampleModulePHP.CalculateAverage” describes that the:

- alias type is custom generated alias,
- if the AliasModuleType is PHP, the PHP class name is ExampleModulePHP,
- and PHP function name is CalculateAverage.

Then, the AF looks in *<service directory>*`\Modules\php` directory for PHP scripts and check if a class was implemented from ICodiacModule interface. For example, the image below refers to ExampleModule.php that contains a module with a class name of ExampleModulePHP that implements ICodiacModule and extends from AFModuleBase. The AFModuleBase is needed so that the class can access the AliasFramework aliases.

ExampleModule.php

```php
<?php

include "afmodulebase.php";

class ExampleModulePHP extends AFModuleBase implements ICodiacModule
{
```

Then, AF loads PHP script with ExampleModulePHP class and executes CalculateAverage function which perform some tasks or calculations.

13.1 Required tools

- Visual Studio Code - https://code.visualstudio.com/
- Visual Studio Code: PHP Debug by Felix Becker

The Codiac.Service\~external directory contains php libraries which are copied to x64\Release and x64\Debug after build. So, you do not install a separate PHP for Codiac service.

13.2 How to create PHP AF extension function?

1. Open service directory in VS Code.
   - service directory is where the Codiac service is
2. Create a new PHP script (press Ctrl-N or go to **File menu → New File**) to *<service directory>*`\Modules\php` directory
3. Create a class that extends from AFModuleBase class and implements ICodiacModule as

shown below.

ExampleModule.php

```php
<?php

include "afmodulebase.php";

class ExampleModulePHP extends AFModuleBase implements ICodiacModule
{
```

4. Create methods that are going to be executed by AF. Below is an example of two empty functions. See examples in Helper functions on how to make tasks or get\set the values of aliases.

```php
class ExampleModulePHP extends AFModuleBase implements ICodiacModule
{
    function Calculation1()
    {
        // some calculations here
    }

    function Task1()
    {
        // some tasks here
    }
```

5. Before the functions can exit, the alias of extension function must be added to evaluated aliases. For Multi Generated alias, set the alias to vector of AliasValue objects. For Custom Generated alias, set the alias to an Alias object. See below for an example**:**

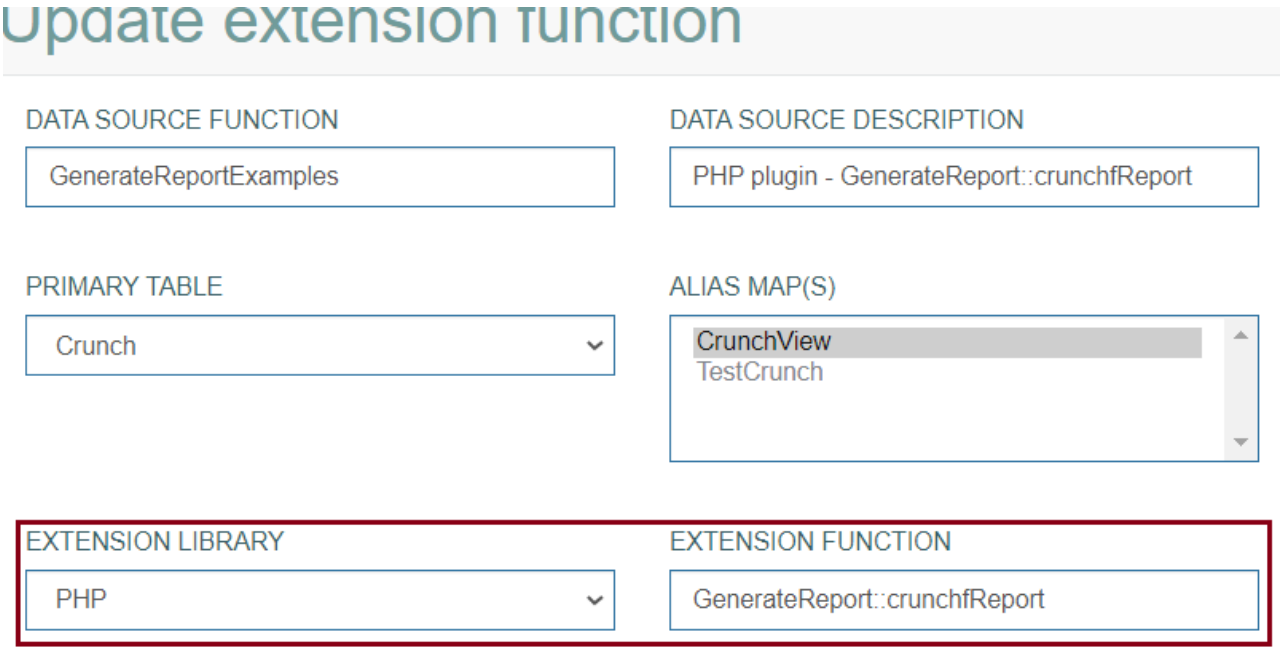
```php
class ExampleModulePHP extends AFModuleBase implements ICodiacModule
{
    function Calculation1()
    {
        // some calculations here

        $outputAliases = array();
        $av1 = new AliasValue();
        $av1->setValue(1.2345);
        $outputAliases[] = $av1;
        $av2 = new AliasValue();
        $av2->setValue(3.1416);
        $outputAliases[] = $av2;

        $this->replaceOrInsertInMultiMap('Multi.ExampleModulePHP.Calculation1', $outputAliases);
    }

    function Task1()
    {
        // some tasks here

        $av = new AliasValue();
        $av->setValue("Completed a tasj");
        $this->replaceOrInsertInMap('Custom.ExampleModulePHP.Task1', $av);
    }
```

6. Finally, add the Custom\Multi Generated alias of your AF extension module in Codiac builder.
    a) Add your Custom\Multi Generated alias to Alias setup
    b) Go to the **Features** → **Management** and click **Reload aliases** button
    c) Use an alias in the screen builder
    d) When loading a screen containing a module alias, the function bound to the alias will be executed.

14 PHP extension functions (plugins)

The extension functions are configured in PHP client. Screenshot below is an example setup which specifies the class and the function to execute. This is different from C++ setup which specifies the user module name.

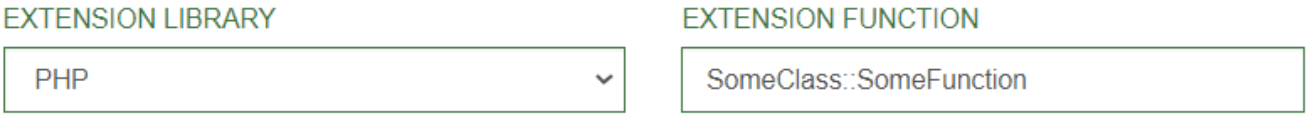
The CodiacSDK.PhpWrapper service handles the loading and execution of PHP script. So, this service is called instead of user module.



Let say, we have “SomePlugin.php” script having a “SomeClass” class with “SomeFunction” method. To set it up as an PHP extension function:

4. In PHP client, go to **System → Extension function → Create**
5. Select “PHP” in **EXTENSION LIBRARY**
6. Put class name and function name in EXTENSION FUNCTION with double colon in between (example: “SomeClass::SomeFunction”)

The setup will look as follows:



Note that the script name (“SomePlugin.php”) is not use in the setup.

14.1 Required tools

- Visual Studio Code - https://code.visualstudio.com/
- Visual Studio Code: PHP Debug by Felix Becker

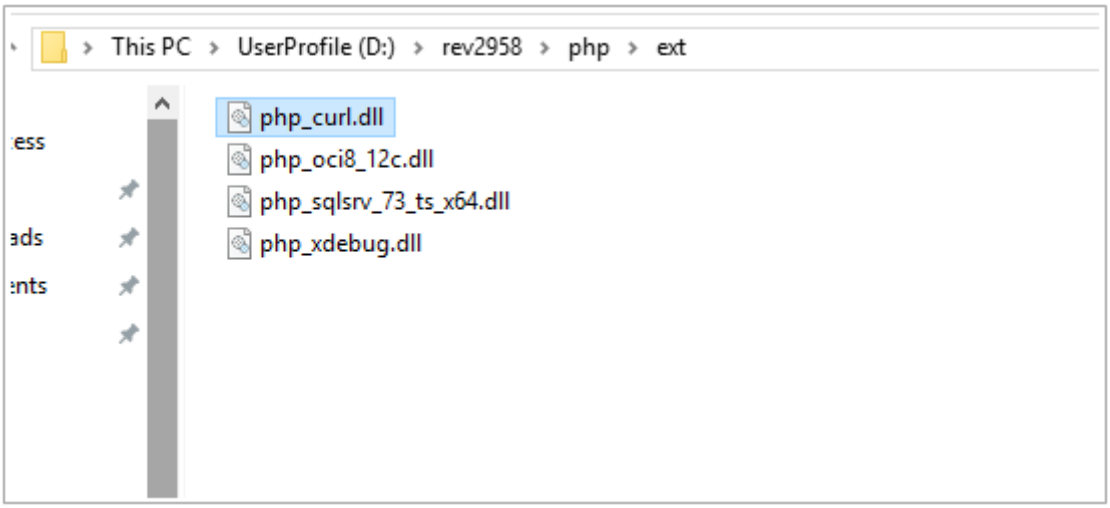
The Codiac.Service\~external directory contains PHP libraries that are copied to x64\Release and x64\Debug after build. So, you do not install a separate PHP for the Codiac service.

14.1.1 PHP additional libraries

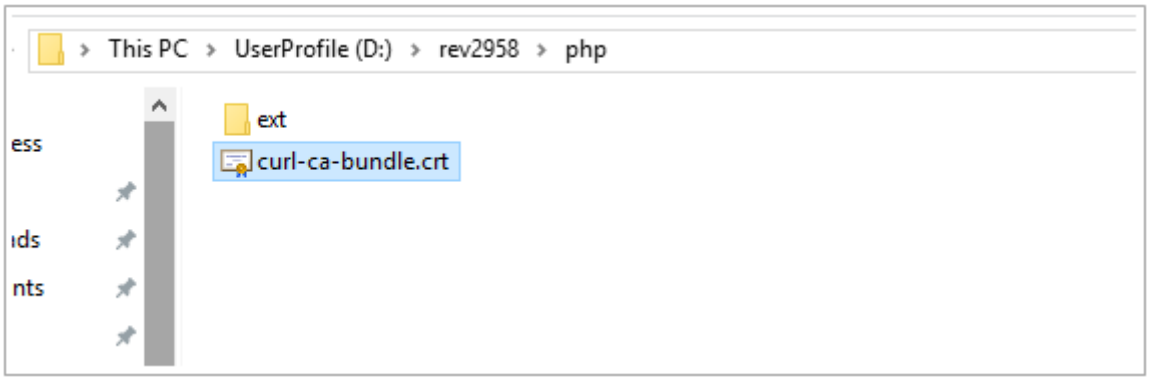
In case the PHP plugins' capabilities should be expanded, additional PHP libraries should be added to the service.

To configure the possibility of using the additional PHP libraries in the service, follow the steps below:

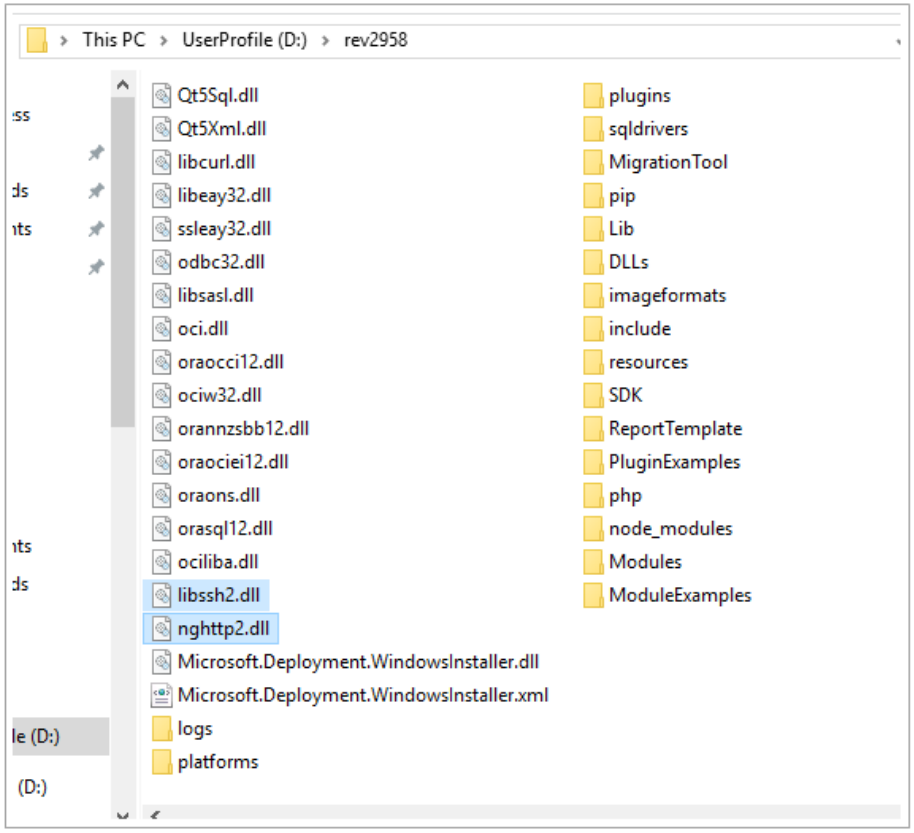
1. Add the library to the *<service_directory>/php/ext* folder.
   For example, to enable the possibility to make an API request for PHP plugins, the **php_curl.dll** library should be added to the service.
   In this example the **php_curl.dll** library is not added during the service installation and should be added as additional one.



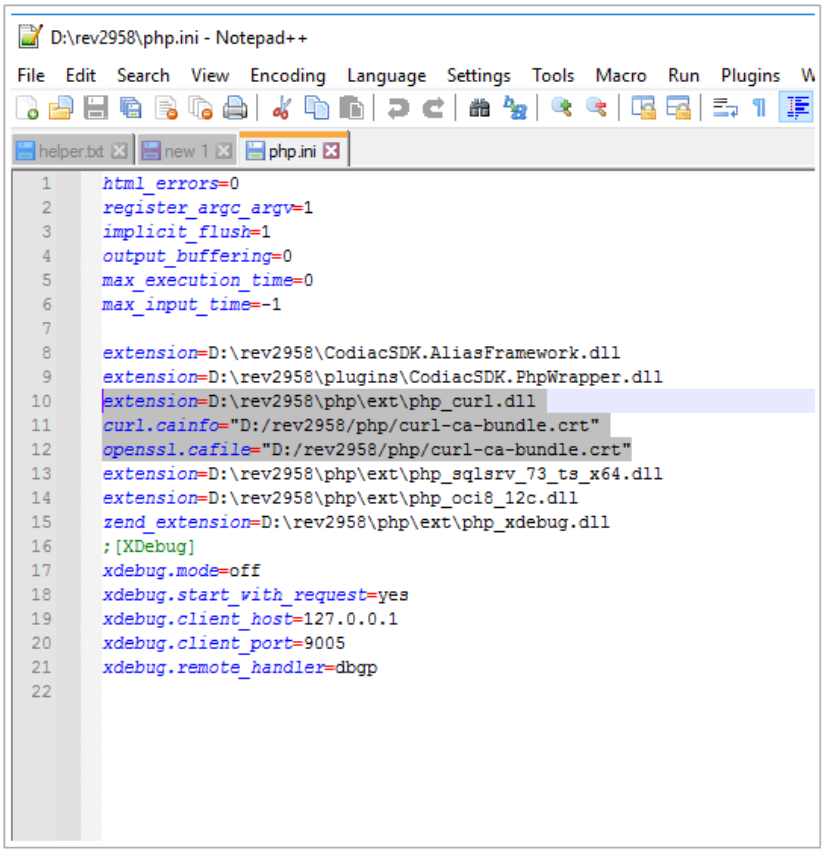
2. For the requests with the **curl** request type, the certificates should be added.
   Add the **curl-ca-bundle.crt** file to the *<service_directory>/php* folder.



3. The additional libraries should be added to the service for the **curl** request type.
   Add the **libssh2.dll** and **nghttp2.dll** libraries to the *<service_directory>* folder.

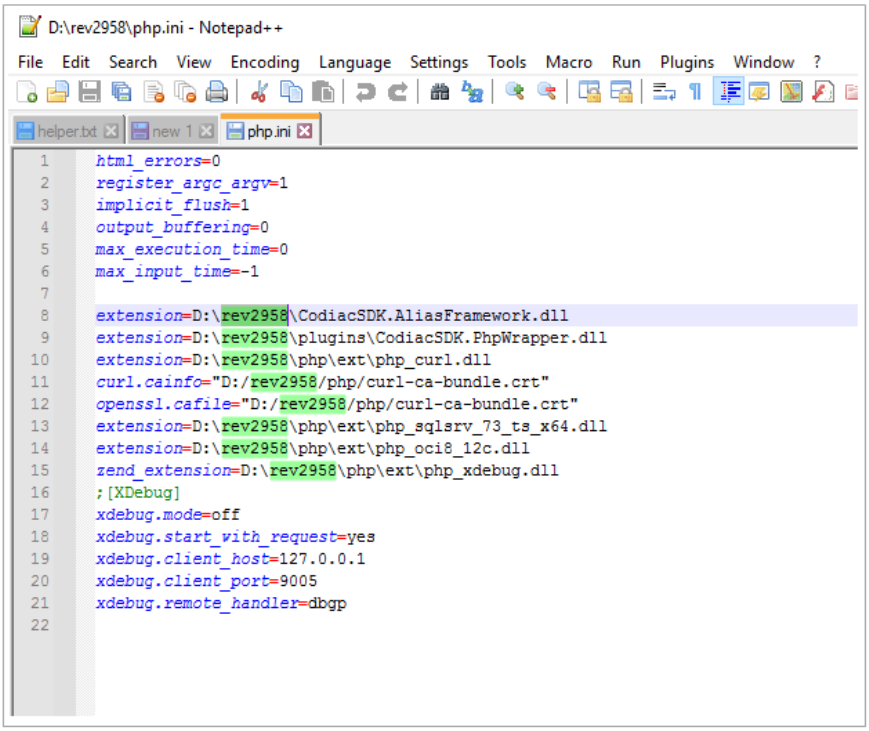


4. To ensure the service recognizes the library and certificates' location, it is necessary to incorporate the path to the **php_curl.dll and curl-ca-bundle.crt** files into the **php.ini** file located in the <*service_directory*> folder.

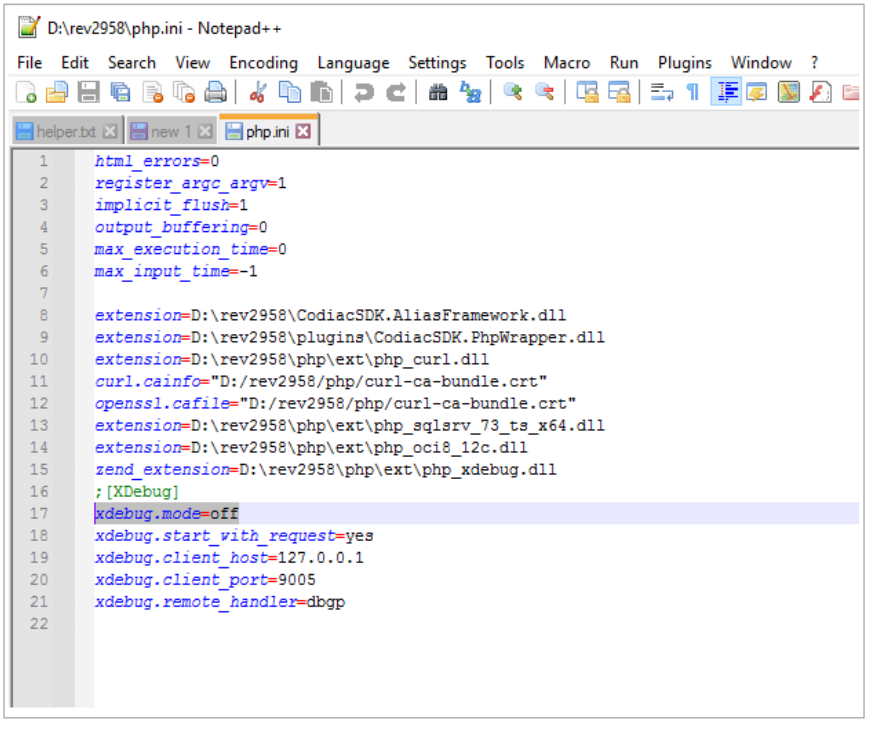


5. Ensure that the service directory is set correctly in the **php.ini** file. As basic configuration is

automatically handled during service installation.



6. The debugging mode can be enabled or disabled in the **php.ini** file.
   Please be aware that enabling debugging mode may significantly decrease the execution speed of plugins.



7. After updating the **php.ini** file and adding extensions, the service is necessary to be restarted.
   Now users can use the **curl** functionality inside plugins.

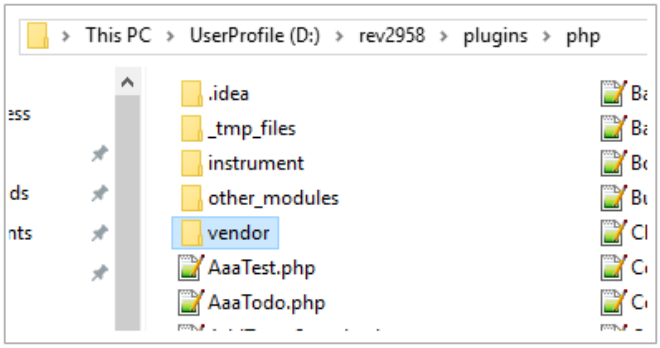
14.1.2 PHP package manager

The additional (third-party) libraries in PHP plugins can be used, for example, in case users need to save data in files containing Excel tables. To streamline this process, the additional libraries should be incorporated for managing **htmlpurifier** tables. To install the **htmlpurifier** library, the Composer package manager is used.

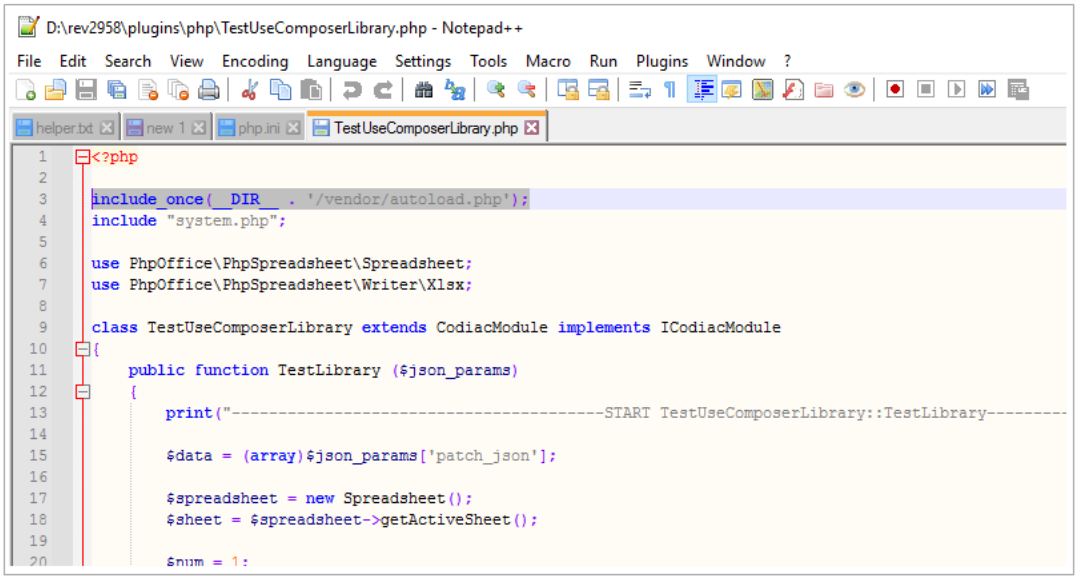
Composer is a package manager that allows users to load third-party libraries written in PHP language.

To configure the possibility of using the additional PHP libraries in the service, follow the steps below:

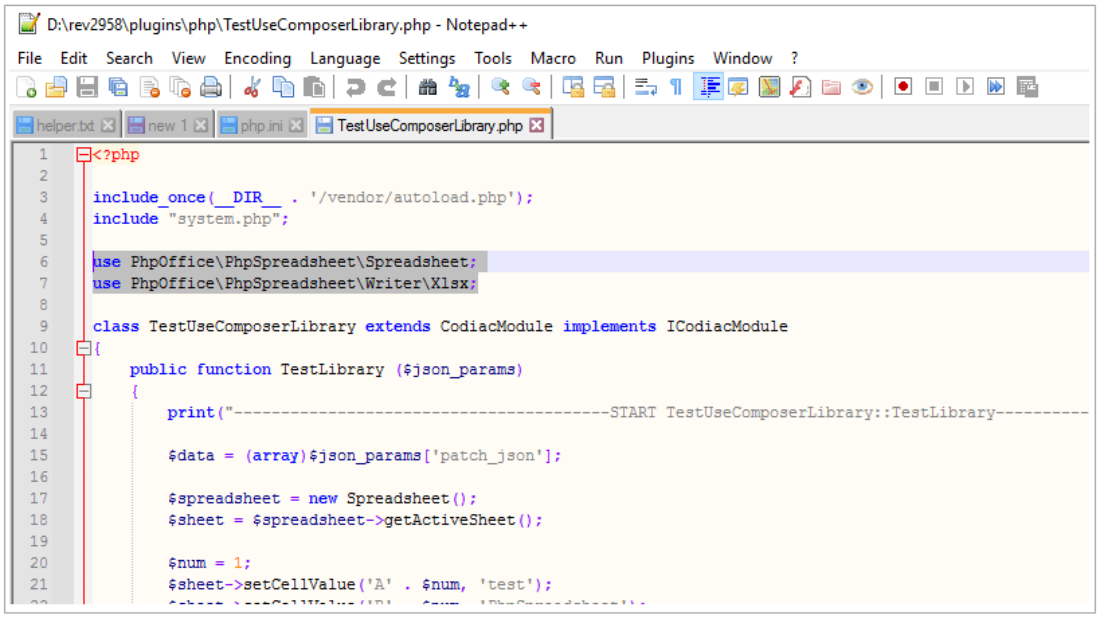
1. Install the **Composer** package manager.
2. Open the **php** folder at the *<service_directory>/plugins/php* path and install the **htmlpurifier** library using Composer.
   If Composer is used to manage dependencies, users can run the command:
   $ composer require ezyang/htmlpurifier
   As a result, the vendor folder appears at the following path:
   <service_directory>/plugins/php/vendor.



3. The **vendor** folder contains the necessary library with all dependencies. If Composer is used to install the additional library, the files will be autoloaded and the namespace will be configured.
   Then, connect the autorun file in the PHP plugin.



In the PHP plugin all necessary classes will be displayed.



4. After adding the third-party libraries and plugin or module files, the service should be restarted.

14.2 Structure of PHP plugins

The PHP plugins structure contains ICodiacModule interface; plugin class which implements ICodiacModule; and CodiacModule which is the base class of plugin class.

- ICodiacModule is an empty interface that is use by CodiacSDK.PhpWrapper to check if it is Codiac module.
- CodiacModule contains the sdk_module_info function and configure function.
  - The sdk_module_info function is executed at start of service. This function returns the list of function in JSON format.
  - The configure function is execute before the plugin function. At the moment, this function configures DataPoolLoader.
- Plugin class contains the plugin function implementation.

14.3 How to create a PHP plugin?

1. Open service directory in VS Code.
   - service directory is where Codiac service is
2. Create a new PHP script (Ctrl-N or go to **File menu → New File**) to *<service directory>*`\plugins\php` directory.
3. Create a PHP class which implements ICodiacModule and extends from CodiacModule class.
4. Create methods for the plugin class.

```
include "system.php";

abstract class MessageType
{
    const NaN = 1;
    const MoreThan400 = 1;
}

class DemoAgentPlugin extends CodiacModule implements ICodiacModule
{
    function CheckYtdPersistCredits() {

        $strValue = getAliasValue("Alias.Demo_Agent.YTDPersistCredits");
        if (!is_numeric($strValue)) {
            return error(MessageType::NaN,
                "YTD Persist Credits of $strValue isn't a number. Please correct this value.");
        }

        $value = floatval($strValue);
        if($value > 400 && !isConfirmed(MessageType::MoreThan400)) {
            return warning(MessageType::MoreThan400,
                "YTD Persist Credits of $strValue is more than 400. Please review. (PHP)");
        }
    }
}
```

5. Finally, add your plugin functions in Codiac builder.
   a) Add your plugin functions to extension function setup.
   b) Configure your screen to include the extension functions.
   c) Execute screen in Codiac render to test your plugin functions.

15 PHP helper functions

These helper functions will allow you to access to Alias Framework, query database, send email, send SMS message, and execute Codiac services.

Note that some functions are only available for AF extension functions, some are only for plugins and other functions are for both.

15.1 AF helper functions for AF extension functions

15.1.1 getFromMap(*aliasName*)

- **Purpose:** return AliasValue object of an alias, or null if not found
- **Example:** This example returns the AliasValue object of "Alias.EmailSendTests.language". With the AliasValue object, the value can be retrieve using valueToString method, getValue method or value property.

```
$avLanguage = $this->getFromMap("Alias.EmailSendTests.language");
if (!is_null($avLanguage)) {
    $language = $avLanguage->valueToString();
```

15.1.2 getFromMultiMap(aliasName)

- **Purpose:** return array of AliasValue objects of multi-value alias, or return null if not found
- **Example:** This example returns the array of AliasValue objects of "Array.CrunchF.Amt". Then, iterate on that list of AliasValue objects for processing each item.

```
$amounts = $this->getFromMultiMap('Array.CrunchF.Amt');
foreach ($amounts as $amount) {
    $amount = $amount->valueToString();
```

15.1.3 replaceOrInsertInMap(aliasName, aliasValue)

- **Purpose:** insert alias (with AliasValue object) in evaluated aliases
- **Example:** This example creates an AliasValue object and add it in evaluated aliases.

```
$av = new AliasValue();
$av->setValue($aliasValue);
$this->replaceOrInsertInMap('Custom.ExampleModulePHP.Email', $av);
```

15.1.4 replaceOrInsertInMultiMap(aliasName, aliasValues)

- **Purpose:** insert multi-value Alias (with vector of AliasValue objects) in evaluated aliases
- **Example:** This example creates list of AliasValue object and add it in evaluated aliases.

```
$outputAliases = array();
$fetchedSql = $sqlResult->fetchOne();
foreach($fetchedSql as $item) {
  $av = new AliasValue();
  $av->setValue($item);
  $outputAliases[] = $av;
}

$this->replaceOrInsertInMultiMap('Multi.ExampleModulePHP.SqlSelectOtherDb', $outputAliases);
```

15.2 AliasValue class

15.2.1 AliasValue constructor

- **Purpose:** create AliasValue object

15.2.2 setValue(value)

- **Purpose:** set AliasValue object with double value, integer value, or string value

15.2.3 getValue(), valueToString()

- **Purpose:** return the string value

15.2.4 value property

- **Purpose:** set or get the string value

15.2.5 markForDeletion()

- **Purpose:** delete database table row pointed by Alias value after execution of AF extension function

Note: See the helper functions for AF for an example usage of AliasValue class.

15.3 AliasView class for PHP plugins

15.3.1 AliasView(aliasView, aliasInput, ruleInput, userInformation, dataPoolLoader) constructor

- **Purpose:** create AliasView instance that evaluates alias view
- **Example:** This example creates AliasView instance and evaluates “CrunchView” alias view with input “Alias.Crunch.Contract” alias having value of “SAMPLEVUL”. The evaluate method stores the evaluated data to the $evaluatedData->returnData argument.

```
$aliasViews = json_encode(array('CrunchView'));
$aliasInput = json_encode(array( 'Alias.Crunch.Contract' => 'SAMPLEVUL' ));
$av = new AliasView($aliasViews, $aliasInput, $funcParam, $funcParam, $this->poolLoader);

$evaluatedData = json_decode($av->evaluate());
if ($evaluatedData->result) {
    $returnData = $evaluatedData->returnData;
```

15.4 JSON request helper functions for PHP plugins

These helper functions parse the JSON request and provide convenient way to retrieve and update the values of aliases. This is are normally use for plugins that going to be assigned as pre-extension functions.

15.4.1 getAliasValue(aliasName)

- **Purpose:** retrieve the string value of an alias
- **Example:** This example retrieve the value of "Alias.Portfolio.Description" alias from JSON request.

```
$aliasVal = getAliasValue("Alias.Portfolio.Description");
```

15.4.2 setAliasValue(aliasName, value)

- **Purpose:** modify the value of an alias
- **Example:** This example modifies "Alias.Portfolio.Description" alias of the JSON request to "new description".

```
setAliasValue("Alias.Portfolio.Description", "new description");
```

15.4.3 getAliasArraySize()

- **Purpose:** returns the size of alias array
- **Example:** This example stores the size of array to variable $arraySize

```
$arraySize = getAliasArraySize();
```

15.4.4 getAliasArrayValue(index, aliasName)

- **Purpose:** retrieve the string value of alias from an array index
- **Example:** This example retrieves the value of each "Array.PortfolioFundsTEST.FundNo" alias from JSON request.

```
$getAliasArrayValue = getAliasArrayValue(3, "Array.PortfolioFundsTEST.FundNo");
```

15.4.5 setAliasArrayValue(index, aliasName, value)

- **Purpose:** modify the value of an alias from an array index
- **Example:** This example replaces the value of "Array.PortfolioFundsTEST.FundNo" with "5".

```
$getAliasArrayValue = setAliasArrayValue(3, "Array.PortfolioFundsTEST.FundNo", "5");
```

15.4.6 getAliases()

- **Purpose:** return aliases (excludes alias array) from values of aliases of JSON request

15.4.7 getAliasArray()

- **Purpose:** return entire alias arrays (excludes non-array aliases) of JSON request

15.5 Message helper functions for PHP plugins

15.5.1 isConfirmed(messageCode)

- **Purpose:** Check if the JSON request has confirmed message_code. This helper function is use with plugin_helpers.INFO and plugin_helpers.WARNING.
- Note: See the examples of plugin_helpers.INFO and plugin_helpers.WARNING for usage of IsConfirmed function.

15.5.2 info(messageCode, message)

- **Purpose:** JSON response with info message
- **Example:** This example sends back a message response to client with an id of 1.

```
if(!isConfirmed(1))
    return info(1, "Starting a task");
```

- **Client output:**

- **Note:** The client will resend the request after it has acknowledged the info message. So, the plugin function must check the messageCode to skip the info message.

15.5.3 warning(messageCode, message)

- **Purpose:** JSON response with warning message

- **Example:** This example sends back a warning message response to client with an id of 2.

```
if(!isConfirmed(2))
    return warning(2, "Test WARNING from PHP");
```

- **Note:** The client will resend the request after it has acknowledge the warning message. So, the plugin function must check the messageCode to skip the warning message.

15.5.4 error(messageCode, message)

- **Purpose:** JSON response with error message
- **Example:** This example sends back an error message response to client.

```
return error(3, "Test ERROR from PHP");
```

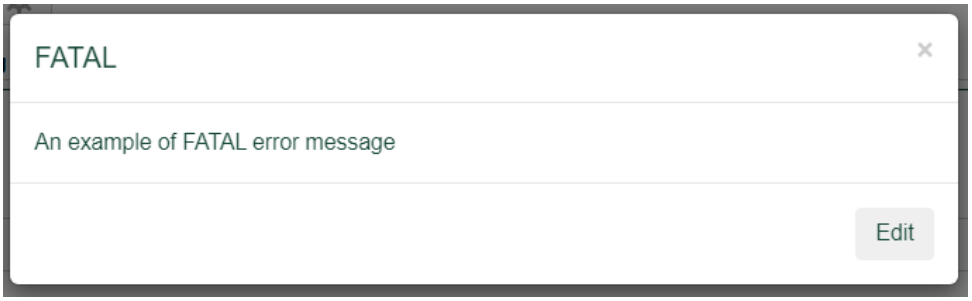
- **Note:** The client doesn't resend the request after it has acknowledge the error message.

15.5.5 fatal(messageCode, message)

- **Purpose:** JSON response with fatal message
- **Example:** This example sends back an fatal message response to client.

```
return fatal(4, "An example of FATAL error message");
```

- **Client output:**



- **Note:** The client doesn't resend the request after it has acknowledge the fatal message.

15.6 Document management helper functions for PHP plugins

15.6.1 copyDocumentToDevice(filePath, fileDescription, documentFamily, documentCategory)

- **Purpose:** Copies document to a specific device, the family and category are optional parameters, if they are not specified the document is copied according to the parameters in the JsonKeeper.
- **Return:** Returns the file container ID if successful, otherwise an empty string is returned.

- **Note:** The example for this function is provided together with the "addDocumentToReturnList" function example.

15.6.2 addDocumentToReturnList(documentID)

- **Purpose:** Adds the file container ID to the JsonKeeper and later in the response of the extension function.
- **Return:** Returns "true" if successfully executed, otherwise "false" is returned.
- **Example:**

```
$isAdded = false;
$fileID = '';
$filePath = 'C:\\DocumentManagementTest.txt';
$fileDescription = 'description';

$documentFamily = 'test';
$documentCategory = 'test';

// copies the document to the device specified on the screen
$fileID = copyDocumentToDevice($filePath, $fileDescription);
// adds the file container ID in the response
$isAdded = addDocumentToReturnList($fileID);
var_dump('File container ID: ' . $fileID . '. Added to return list: ' . $isAdded . '.');

// copies the document to the device specified in the parameters
$fileID = copyDocumentToDevice($filePath, $fileDescription, $documentFamily, $documentCategory);
// adds the file container ID in the response
$isAdded = addDocumentToReturnList($fileID);
var_dump('File container ID: ' . $fileID . '. Added to return list: ' . $isAdded . '.');

return array('resultbody'=> array('requestresult'=> 'successfully', 'extendedinfo'=> 'OK'));
```

15.6.3 getDocumentFromDevice(fileContainerID, filePath)

- **Purpose:** Loads the document with the specified file container ID in the specified path.
- **Return:** Returns "true" if successfully executed, otherwise "false" is returned.
- **Example:**

```
$isLoaded = false;
$fileID = '1111';
$filePath = 'D:\\Users\\spoposki\\Desktop\\testFile.jpg';

// loads the document with the specified file container ID in the specified path
$isLoaded = getDocumentFromDevice($fileID, $filePath);
var_dump('File container ID: ' . $fileID . '. Is loaded: ' . $isLoaded . '. Location: ' . $filePath);

return array('resultbody'=> array('requestresult'=> 'successfully', 'extendedinfo'=> 'OK'));
```

15.6.4 deleteDocumentFromDevice(fileContainerID, performHardDelete)

- **Purpose:** Sets the file container "Active" flag to "false" or deletes the file container

completely, it depends on the value of the “performHardDelete” parameter.

- **Returns:** Returns “true” if successfully executed, otherwise “false” is returned.
- **Example:**

```
$isDeleted = false;
$fileID = '1111';

// performs a soft delete on the specified file container
$isDeleted = deleteDocumentFromDevice($fileID, false);
var_dump('File container ID: ' . $fileID . '. Is soft deleted: ' . $isDeleted);

// performs a complete delete on the specified file container
$isDeleted = deleteDocumentFromDevice($fileID, true);
var_dump('File container ID: ' . $fileID . '. Is completely deleted: ' . $isDeleted);

return array('resultbody'=> array('requestresult'=> 'successfully', 'extendedinfo'=> 'OK'));
```

15.6.5 searchForDocument(documentFamily, documentCategory, kp1, kp2, kp3, kp4, kp5, description)

- **Purpose:** Searches for file containers using the function parameters as search parameters. If a function parameter is an empty string that field will not be included in the search.
- **Return:** Returns serialized JSON array containing information about the file containers that match the search parameters.
- **Example:**

```
$fileRootPath = 'D:\\Users\\spoposki\\Desktop\\';
$kp1 = 'sally';
$description = 'test1';

// searches for the documents that have the value "test" in description
$files = searchForDocument('', '', '', '', '', '', '', $description);
var_dump('Files: ' . $files);
// parses the response and iterates through the array
$filesArray = json_decode($files, true);
foreach ($filesArray as $file) {
    $isLoaded = false;
    $filePath = $fileRootPath . $file['container_file_name'];
    // loads the document to disk
    $isLoaded = getDocumentFromDevice(strval($file['id']), $filePath);
    var_dump('File container ID: ' . $file['id'] . '. Is loaded: ' . $isLoaded . '. Location: ' . $filePath);
}

return array('resultbody'=> array('requestresult'=> 'successfully', 'extendedinfo'=> 'OK'));
```

15.6.6 setDocumentKeyParts(fileContainerID, kp1, kp2, kp3, kp4, kp5)

- **Purpose:** Sets the file container key parts.
- **Return:** Returns “true” if successfully executed, otherwise “false” is returned.
- **Example:**

```
$isSet = false;

$fileID = '1111';
$kp1 = 'test_kp1';
$kp2 = 'test_kp2';
$kp3 = 'test_kp3';
$kp4 = 'test_kp4';
$kp5 = 'test_kp5';

// sets the key parts
$isSet = setDocumentKeyParts($fileID, $kp1, $kp2, $kp3, $kp4, $kp5);
var_dump('Key parts set: ' . $isSet);

return array('resultbody'=> array('requestresult'=> 'successfully', 'extendedinfo'=> 'OK'));
```

15.7 Database helper functions for PHP plugins and PHP AF extension functions

15.7.1 DBCursor

- **Purpose:** store the query results of execute_sql; and able to fetch row sequentially from that query results
- **Methods:**
  - size() returns number of rows
  - setPosition(position) set the position of cursor in query results
  - getPosition() return the position of cursor in query results
  - fetchOne() returns row (in list of columns); and next fetch_one() call returns next row. Note that the column names are in alphabetic order.
  - columnNames() returns the column names of query results. The column names are in alphabetic order.
- **Note:** See the example of executeSQL function for usage

15.7.2 executeSQL(sqlQuery, dbConnector)

- **Purpose:** query the database by running SQL query to Codiac database or to other database (with DB connection information)
- **Note:** dbConnector parameter is optional and default to empty which means to connect to Codiac database
- **Example:** This example run SQL query to PostgreSQL database (with DB connection information) and dump the first row result.

```
$dbConnector = new DbConnector("Postgres", "[redacted]", "5432", "[redacted]", "postgres", "[redacted]");
$sqlResult = executeSQL("select * from \"PolicyAgent\"", $dbConnector);

var_dump("DB Cursor size");
var_dump($sqlResult->size());

var_dump("DB fetch one");
var_dump($sqlResult->fetchOne());
```

15.8 Other CodiacSDK helper functions for PHP plugins and PHP AF extension functions

15.8.1 sendEmail(to, subject, body)

- **Purpose:** send an email to recipient with subject and message
- **Example:** This AF extension function example sends an email with recipient from "Alias.EmailSendTests.recipient" alias, subject from "Alias.EmailSendTests.subject" alias and message from "Alias.EmailSendTests.body" alias.

```
$avRecipient = $this->getFromMap("Alias.EmailSendTests.recipient");
$recipient = $avRecipient->valueToString();
$avSubject = $this->getFromMap("Alias.EmailSendTests.subject");
$subject = $avSubject->valueToString();
$avBody = $this->getFromMap("Alias.EmailSendTests.body");
$body = $avBody->valueToString();
sendEmail($recipient, $subject, $body);
```

15.8.2 sendEmail(to, subject, body, attachments)

- **Purpose:** Sends an email to recipient with subject, message and attachments. The attachment parameter is a delimited string of file container IDs.
- **Return:** A serialized JSON with details.
- **Example:**

```
$to = 'slavkopoposki@gmail.com';
$subject = 'SUBJECT';
$body = 'BODY';

$files = '1178;1179;1180';

// sends an email with attachments, the file that correspond to the id will the attached to the email
sendEmail($to, $subject, $body, $files);

return array('resultbody'=> array('requestresult'=> 'successfully', 'extendedinfo'=> 'OK'));
```

15.8.3 sendSms(to, body)

- **Purpose:** send SMS message to recipient with message
- **Example:** This example send SMS message to 5556789090 (not a real phone number). Only US numbers are accepted.

```
sendSms("5556789090", "This is a test message");
```

15.8.4 generateReport(reportName, inlineJson, isSaveToXml)

- **Purpose:** generate report PDF file from report template (client report builder)
- Note: The isSaveToXml default to false; if set to true, the report are save to XML file and is not generated to PDF.
- **Example:** This is example generate reports from alias view data. The alias view data contains alias values in JSON format.

```
$aliasViewData = $jsonParams['alias_view_data'];
$inlineJson = '';
if (is_array($aliasViewData) && !empty($aliasViewData)) {
    $inlineJson = json_encode($aliasViewData[0]);
} else {
    $inlineJson = json_encode($aliasViewData);
}

$reportName = 'Test CrunchF Report';
return $this->generateReportInlineJson($jsonParams, $reportName, $inlineJson);
```

- **Example:** Another example that generate report from report template “Test CrunchF Report” with values from dependent multi-value Aliases. Note that the function has to create a JSON input string from aliases.

```
// dependency aliases
$contracts = $this->getFromMultiMap('Array.CrunchF.Contract');
$descriptions = $this->getFromMultiMap('Array.CrunchF.Description');
$fundNos = $this->getFromMultiMap('Array.CrunchF.FundNo');
$amounts = $this->getFromMultiMap('Array.CrunchF.Amt');

$jsonInput = array(
    'Array.CrunchF' => array()
);

for ($i = 0; $i < count($contracts); $i++) {
    $item = array(
        'Array.CrunchF.Contract' => $contracts[$i]->valueToString(),
        'Array.CrunchF.Description' => $descriptions[$i]->valueToString(),
        'Array.CrunchF.FundNo' => $fundNos[$i]->valueToString(),
        'Array.CrunchF.Amt' => $amounts[$i]->valueToString()
    );
    $jsonInput['Array.CrunchF'][] = $item;
}

// generate report
generateReport('Test CrunchF Report', json_encode($jsonInput));
```

15.8.5 generateReport(reportName, inlineData, delimiter, isSaveToXml)

- **Purpose:** generate report PDF file from report template (client report builder)
- Note: The isSaveToXml default to false; if set to true, the report are save to XML file and is not generated to PDF.
- **Example:** This example generate report from report template “Test CrunchF Report” with values from vector of delimited values. The delimiter used '|'.

```
$inlineData = array();
$inlineData[] = 'Contract|FundNo|Description|Amount';
$inlineData[] = 'Contract A|CA1|Test contract A - 1|123.45';
$inlineData[] = 'Contract A|CA2|Test contract A - 2|3.45';
$inlineData[] = 'Contract B|CB1|Test contract B - 1|645';
$inlineData[] = 'Contract C|CC1|Test contract C - 1|7.89';

// generate report
generateReport('Test CrunchF Report - inline data', $inlineData, '|');
```

15.8.6 executeJson(jsonRequest)

- **Purpose:** call Codiac service by executing JSON request
- **Example:** This example calls JobsScheduler::GetJobsScheduleList to get list of jobs. A JSON request is created with func_name parameter and lib_name parameter.

```
// create json request
$jsonRequest = array(
    'requestbody' => array(
        'func_name' => 'GetJobScheduleList',
        'lib_name' => 'CodiacSDK.zJobsScheduler',
    )
);

// execute service to get Jobs
$jsonResponse = executeJson(json_encode($jsonRequest));
```

- Note: The user\session parameters are filled up at service process.

15.9 SQL query with PHP native database

Codiac SDK provides database helper functions that can access Codiac database and other databases with connection string. In addition, the user can directly call the PHP native library to query the database.

The ExampleModule AF extension module contain example source code on how to query the databases with native libraries.

- **PostgreSQL**

Below is the sequence for PHP PostgreSQL to query database:

1. Call “pg_connect” function with connection string to connect to PostgreSQL database
2. Call “pg_query” function with connection object to execute a query
    - returns query result object
3. Call “pg_fetch_row” function with query result object to fetch query result row
4. Call “close” function on connection object to close connection to database

See the PostgreSQL function in ExampleModule.php for full example.

See the https://www.php.net/manual/en/book.pgsql.php for full details on how to use PostgreSQL.

- **MySQL**

  Below is the sequence for PHP MySQL to query database:

  1. Call “mysqli_connect” function with connection information to connect to MySQL database
  2. Call “mysqli_query” function with connect object to execute a query
  3. Call “prepare” function on connect object to prepare the query which return statement object
  4. Call “execute” function on statement object to execute the query
  5. Call “bind_result” function to bind the variables to column results
  6. Call “fetch” function to fetch the column results to bound variables
  7. Call “close” function on statement object
  8. Call “close” function on connect object to close the connection to database

  See the MySQL function in ExampleModule.php for full example.

  See the https://www.php.net/manual/en/book.mysqli.php for full details on how to use MySQL Improved extension.

- **MS SQL Server**

  To query MS SQL Server, the php.ini (located in service root directory) must include `php_sqlsrv_73_ts_x64.dll` in extension list as shown below:

php.ini

```
1  html_errors=0
2  register_argc_argv=1
3  implicit_flush=1
4  output_buffering=0
5  max_execution_time=0
6  max_input_time=-1
7  extension=C:\Codiac.Service\x64\Release\CodiacSDK.AliasFramework.dll
8  extension=C:\Codiac.Service\x64\Release\plugins\CodiacSDK.PhpWrapper.dll
9  extension=C:\Codiac.Service\x64\Release\php\ext\php_sqlsrv_73_ts_x64.dll
10 extension=C:\Codiac.Service\x64\Release\php\ext\php_oci8_12c.dll
11 zend_extension=C:\\Codiac.Service\x64\Release\php\ext\php_xdebug.dll
12 ;[XDebug]
13 xdebug.mode=debug
14 xdebug.start_with_request=yes
15 xdebug.client_host=127.0.0.1
16 xdebug.client_port=9005
17 xdebug.remote_handler=dbgp
```

Then. install ODBC driver 17 for SQL Server: https://docs.microsoft.com/en-us/sql/connect/odbc/download-odbc-driver-for-sql-server

Below is the sequence for PHP MS SQL Server to query database:

1. Call “sqlsrv_connect” function with connection information to connect to MS SQL Server database
2. Call “sqlsrv_query” function with connect object to execute a query
   - returns query result object
3. Call “sqlsrv_fetch_array” function with query result object to fetch query result row
4. Call “sqlsrv_free_stmt” function to free up the query result object
5. Call “close” function on connect object to close the connection to database

See the MsSqlServer function in ExampleModule.php for full example.

See the https://www.php.net/manual/en/book.sqlsrv.php for full details on how to use MS SQL Server Driver for PHP.

- **Oracle**

  To query Oracle SQL Server, the php.ini (located in service root directory) must include Oracle OCI8 extension (i.e. `php_oci8_12c.dll`) in extension list as shown below:

php.ini

```
1   html_errors=0
2   register_argc_argv=1
3   implicit_flush=1
4   output_buffering=0
5   max_execution_time=0
6   max_input_time=-1
7   extension=C:\Codiac.Service\x64\Release\CodiacSDK.AliasFramework.dll
8   extension=C:\Codiac.Service\x64\Release\plugins\CodiacSDK.PhpWrapper.dll
9   extension=C:\Codiac.Service\x64\Release\php\ext\php_sqlsrv_73_ts_x64.dll
10  extension=C:\Codiac.Service\x64\Release\php\ext\php_oci8_12c.dll
11  zend_extension=C:\\Codiac.Service\x64\Release\php\ext\php_xdebug.dll
12  ;[XDebug]
13  xdebug.mode=debug
14  xdebug.start_with_request=yes
15  xdebug.client_host=127.0.0.1
16  xdebug.client_port=9005
17  xdebug.remote_handler=dbgp
```

Below is the sequence for PHP Oracle SQL to query database:

1. Call “oci_connect” function with connection information to connect to Oracle SQL Server database
2. Call “oci_parse” function with SQL command to prepare the query for execution
3. Call “oci_execute” function with connect object to execute a query
4. Call “oci_fetch_array” function with statement id to fetch query result row
5. Call “oci_free_statement” function to free up the query result object
6. Call “oci_close” function on connect object to close the connection to database

See the OracleSQL function in UnitValue.php and ExampleModule.php for full example.

See the https://www.php.net/manual/en/book.oci8.php for full details on how to use Oracle OCI8 for PHP.

16 PHP Debugging with Visual Studio Code

Before starting the debugger, create launch.json file for PHP debugging. This settings means that VS Code will listen to port 9005 at localhost for XDebug.

```
{
  // Use IntelliSense to learn about possible attributes.
  // Hover to view descriptions of existing attributes.
  // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Listen for XDebug",
      "type": "php",
      "request": "launch",
      "port": 9005,
      "hostname": "localhost"
    }
  ]
}
```

XDebug extension needs to be added and configured in php.ini which is located in service root directory. Below are the default settings:

php.ini

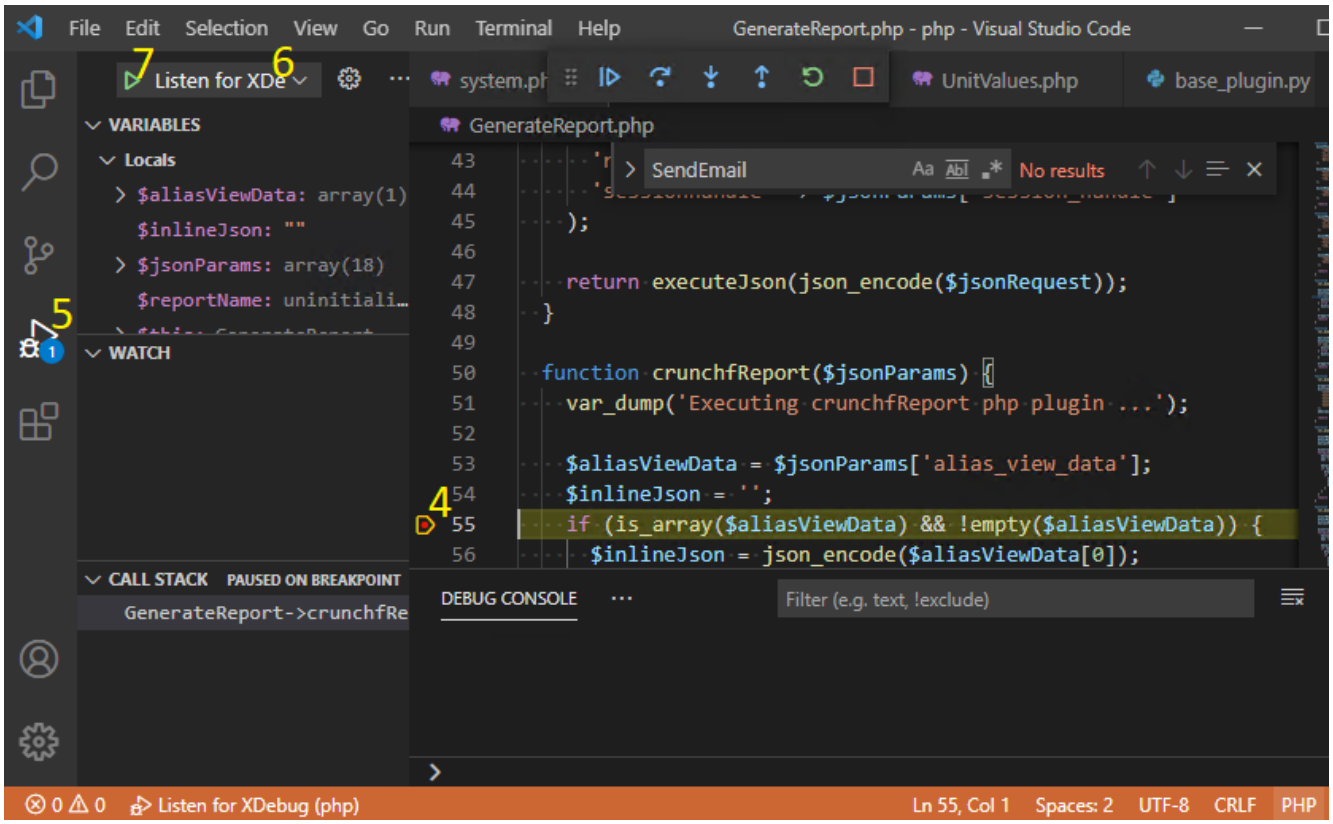
```
1   html_errors=0
2   register_argc_argv=1
3   implicit_flush=1
4   output_buffering=0
5   max_execution_time=0
6   max_input_time=-1
7   extension=C:\Codiac.Service\x64\Release\CodiacSDK.AliasFramework.dll
8   extension=C:\Codiac.Service\x64\Release\plugins\CodiacSDK.PhpWrapper.dll
9   extension=C:\Codiac.Service\x64\Release\php\ext\php_sqlsrv_73_ts_x64.dll
10  extension=C:\Codiac.Service\x64\Release\php\ext\php_oci8_12c.dll
11  zend_extension=C:\Codiac.Service\x64\Release\php\ext\php_xdebug.dll
12  ;[XDebug]
13  xdebug.mode=debug
14  xdebug.start_with_request=yes
15  xdebug.client_host=127.0.0.1
16  xdebug.client_port=9005
17  xdebug.remote_handler=dbgp
```

For debugging PHP script,

- php_xdebug.dll is in the extension list
- xdebug.mode must be set to debug
- client_host and client_port matched the VS Code launch.json file.

While the service is running, starts VS Code debugger:

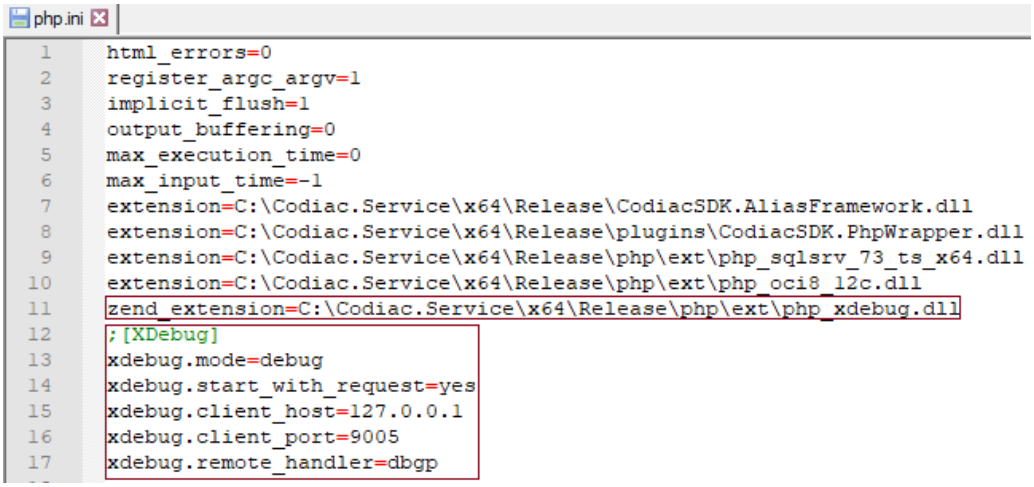
1. Run the VS Code in administrator account
2. Open the PHP directory to be debugged
   - AF extension function: x64\<*Debug*|*Release*>\Modules\php.
   - plugins: x64\<*Debug*|*Release*>\plugins\php.
3. Open the PHP script to be debugged.
4. Add breakpoint by clicking the left side of line number.
5. Click the debug icon on left side bar.
6. Select the Listen for XDebug.
7. Click the green play button (or F5) to start debugging. At this point, VS Code is attached to the debugger.
8. Execute the PHP plugin or PHP AF extension function.
9. Wait for a few seconds to stop at breakpoint.



10. You can step through the codes by pressing the F10 (step next), F11 (step into), and F5 (continue) keys or stop (orange square icon). You can add watches or see local watches.

17 Remote Debugging PHP plugin and AF extension functions

In the PHP language, remote debugging is almost identical to local debugging.
To set the remote debugging PHP plugin and AF extension functions, configure the **php.ini** file which is located in service root directory. Below are the default settings:



For debugging a PHP script, the following parameters should be set:

- php_xdebug.dll is in the extension list.
- xdebug.mode must be set to debug.
- client_host and client_port matched the VS Code launch.json file.

The IP address (client_host) and the port (client_port) in the **php.ini** file should be visible and accessible from the outer machine.
Then, perform the debugging steps as for the local debugging described above in the Debugging section.

18 JavaScript Alias Framework extension module

The Alias Framework (AF) service can evaluate custom generated aliases or multi generated aliases. This is done by going thru the alias name structure:

AliasType.ModuleName.FunctionName

Depending on the AliasModuleType, the custom alias can be C++ function, Python function, JavaScript function or PHP function.

For example, the "Custom.ExampleModuleJS.CalculateAverage" tells us that the:

- alias type is custom generated alias,
- if the AliasModuleType is JS, the JavaScript class name is ExampleModuleJS,
- and JavaScript function name is CalculateAverage.

Then, AF spawns ScriptExecutorsJs.exe with the JavaScript class name and function name. The ScriptExecutorJS process looks for matching JavaScript class name and function name in *Modules\js* directory. Finally, loads the class and execute the function.

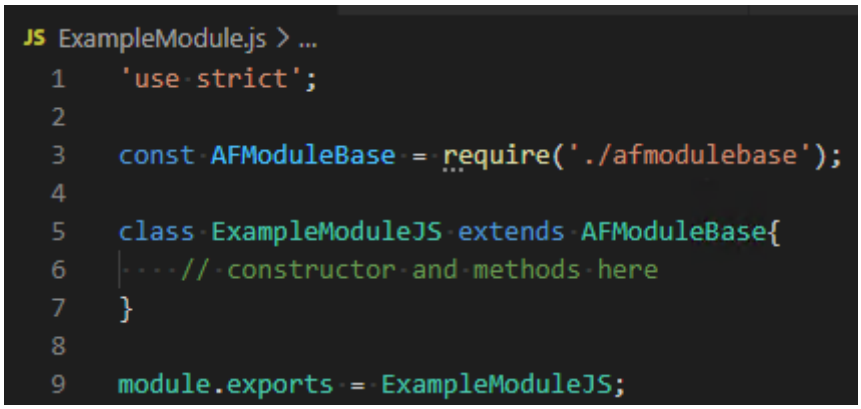
18.1 Required tools

- Visual Studio Code - https://code.visualstudio.com/

At the moment, only Visual Studio Code application is needed for JavaScript AF extension function.

18.2 How to create JavaScript AF extension function?

1. Open service directory in VS Code.
2. Create a new JavaScript script (Ctrl-N or go to **File menu** → **New File**) to *<service directory>*`\Modules\js` directory.
3. Create a class that extends from AFModuleBase class and add the class name to module exports at end of file as shown below.



4. Create methods that are going to be executed by AF. Below is an example of two empty

functions. See examples in Helper functions on how to make tasks or get\set the values of aliases.

```
class ExampleModuleJS extends AFModuleBase {
    Calculation1() {
        // some calculations here
    }

    Task1() {
        // some tasks here
    }
}
```

5. Before the functions can exit, the alias of extension function must be added to evaluated aliases.
   - For Multi Generated alias, set the alias to vector of AliasValue objects.
   - For Custom Generated alias, set the alias to an Alias object.

   See below for an example:

```
class ExampleModuleJS extends AFModuleBase {

    Calculation1() {
        // some calculations here

        let values = [];

        let av1 = new module.AliasValue();
        av.setValueDouble(1.2345);
        values.push(av1);

        let av2 = new module.AliasValue();
        av2.setValueDouble(3.1416);
        values.push(av2);

        this.insertValueInMultiMap("Multi.ExampleModuleJS.Calculation1", values);
    }

    Task1() {
        // some tasks here

        let av = new module.AliasValue();
        av.setValueString("Completed a task");
        this.insertValueInMap("Custom.ExampleModuleJS.Task1", av);
    }
}
```

6. Finally, add the Custom\Multi Generated alias of your AF module in Codiac builder.
   a) Add your Custom\Multi Generated alias to Alias setup
   b) Go to the page **Features → Management** and click **Reload aliases** button

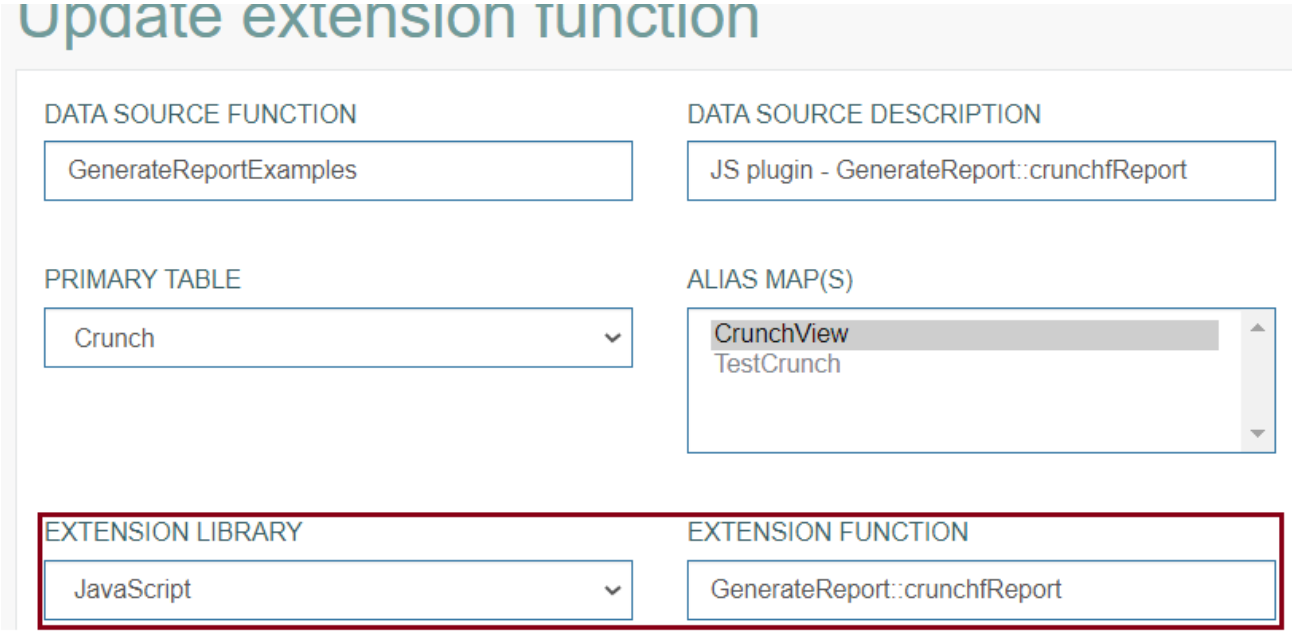
c) Use an alias in the screen builder.

d) When loading a screen containing a module alias, the function bound to the alias will be executed.

19 JavaScript extension functions (plugins)

The extension functions are configured in PHP client. Screenshot below is an example setup which specifies the class and the function to execute. This is different from C++ setup which specifies the user module name.

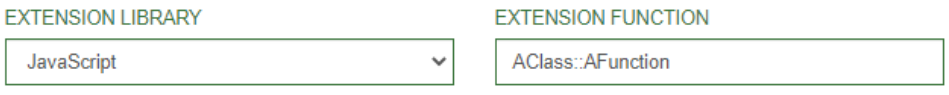
The CodiacSDK.JavaScriptWrapper service handles the loading and execution of JavaScript script. So, this service is called instead of user module.



Let say, we have “APlugin.js” script having a “AClass” class with “AFunction” method. To set it up as an JavaScript extension function,

1. In PHP client, go to **System → Extension function → Create**
2. Select “JavaScript” in **EXTENSION LIBRARY**
3. Put class name and function name in EXTENSION FUNCTION with double colon in between (example: “AClass::AFunction”)

The setup will look like below. Note that the script name (“APlugin.js”) is not use in the setup.



19.1 Required tools

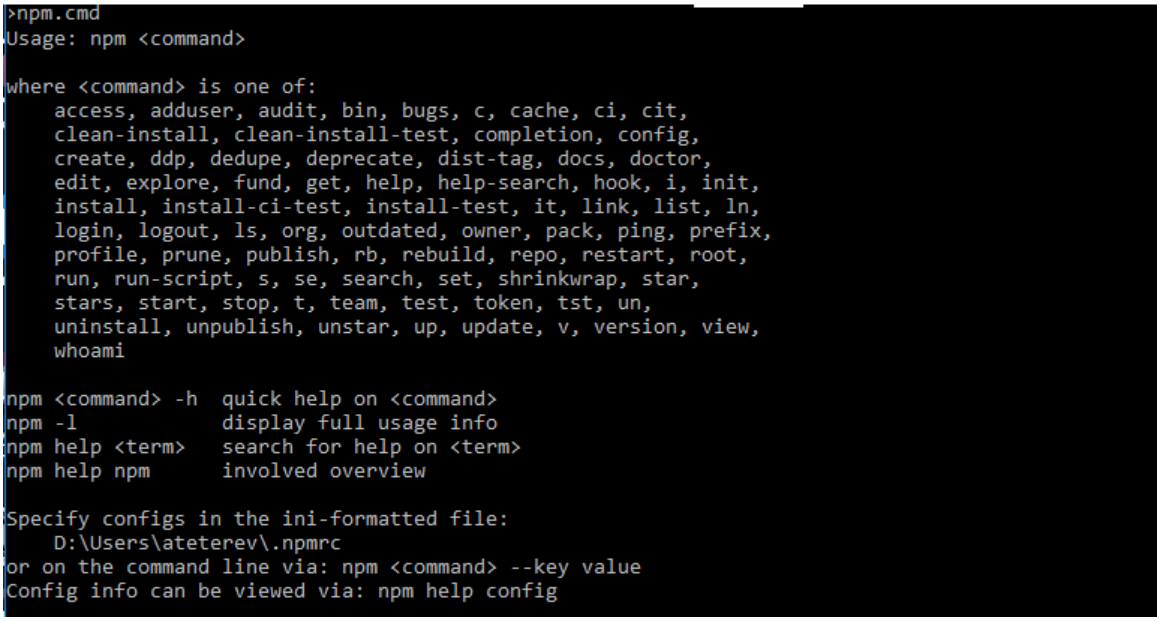
- Visual Studio Code - https://code.visualstudio.com/

At the moment, only Visual Studio Code application is needed for JavaScript plugins.

19.1.1 JavaScript package manager

npm (Node Package Manager) is a default package manager for the JavaScript, primarily used with

the Node.js runtime environment. npm allows developers to easily install, manage, and share reusable code packages and dependencies for their JavaScript projects. It provides a vast ecosystem of packages that developers can leverage to accelerate their development process. Additionally, npm facilitates version management, dependency resolution, and package publishing, making it an essential tool in the JavaScript ecosystem.
If the Codiac service is installed with the JavaScript components, the package manager will be placed in the root folder and have the **npm.cmd** file name.



To use npm in the Codiac service, perform the following actions:

- Launch the Command Prompt utility.
- Use **Help** to see a list of tips that relate to npm. For that, type **npm <command>** and press the Enter key. The list will be shown.
- In the opened list, users can see usage tips, commands, and general options examples. For example, if it is necessary:
    - to see the list of the installed packages, type the following command in the Command Prompt:
      npm.cmd ls
    - to search needed package, type the following command in the Command Prompt:
      npm search crypto-js
    - to install package, type the following command in the Command Prompt:
      npm install crypto-js

All installed packages will be placed in the folder **node_modules**.
More details about supported commands can be found at the following link:
https://docs.npmjs.com/cli/v6/commands.

19.2 Structure of JavaScript plugin

The JavaScript plugins structure contains a JavaScript plugin script and system.js.

- Plugin script contains the plugin class\function implementation
- system.js contains the sdk_module_info function
    - The sdk_module_info function is executed at start of service. This function returns the

list of function in JSON format.

19.3 How to create a JavaScript plugin?

1. Open service directory in VS Code.
   - service directory is where Codiac service is
2. Create a new JavaScript script (Ctrl-N or go to **File menu → New File**) to *<service directory>*\plugins\js directory
3. Create a JavaScript class and require the system.js as shown below

```
JS ExamplePlugin.js > ...
1   'use strict';
2
3   const system = require('./system.js');
4
5   class ExamplePlugin {
6
7   }
8
9   module.exports = ExamplePlugin
```

4. Create methods for the plugin class (see the example of JavaScript helper functions on how to add implementations to plugin functions)

```
1   'use strict';
2
3   const system = require('./system.js');
4
5   class JsTestPlugin {
6     constructor(cfg) {
7
8       //debugger;
9       this.cfg = cfg;
10    }
11
12  > ShuffleComments(json_params) {···
51    }
52
53  > StringShuffle(str) {···
64    }
65
66    sdk_module_info() {
67      return system.sdk_module_info(this);
68    }
69  }
70
71  module.exports = JsTestPlugin
72
```

The plugin class must contain the sdk_module_info() method (shown below):

```
sdk_module_info() {
    return system.sdk_module_info(this);
}
```

5. Finally, add your plugin functions in Codiac builder.
    a) Add your plugin functions to extension function setup
    b) Configure your screen to include the extension functions
    c) Execute screen in Codiac render to test your plugin functions

20 JavaScript helper functions

These helper functions will allow you to access to Alias Framework, query database, send email, send SMS message, and execute Codiac services.

Note that some functions are only available for AF extension functions, some are only for plugins and other functions are for both.

20.1 AF helper functions for AF extension functions

This helpers functions are defined in base class (AFModuleBase). So, these functions can be called with “this” instance.

20.1.1 getValueFromMap(*aliasName*)

- **Purpose:** return AliasValue object of an alias, or null if not found
- **Example:** This example returns the AliasValue object of “Alias.EmailSendTests.language”. With the AliasValue object, the value can be retrieve using valueToString method.

```
let avLanguage = this.getValueFromMap("Alias.EmailSendTests.language");
const language = avLanguage.valueToString();
```

20.1.2 getValueFromMultiMap(aliasName)

- **Purpose:** return array of AliasValue objects of multi-value alias, or return null if not found
- **Example:** This example returns the array of AliasValue objects of “Array.CrunchF.Amt”. Then, iterate on that list of AliasValue objects by index.

```
const contracts = this.getValueFromMultiMap('Array.CrunchF.Contract');
for (let i = 0; i < contracts.length; i++)
{
    const item = {
        'Array.CrunchF.Contract': contracts[i].valueToString(),
```

20.1.3 replaceOrInsertValueInMap(aliasName, aliasValue)

- **Purpose:** insert alias (with AliasValue object) in evaluated aliases
- **Example:** This example creates an AliasValue object and add it in evaluated aliases.

```
this.insertValueInMap('Custom.ExampleModuleJS.ExternalCalculation', av)
```

20.1.4 replaceOrInsertValueInMultiMap(aliasName, aliasValues)

- **Purpose:** insert multi-value Alias (with vector of AliasValue object) in evaluated aliases
- Example:This example creates list of AliasValue objects and add it in evaluated aliases.

```
let outputAliases = [];
const fetchedSql = sqlResult.fetchOne();
for (let i = 0; i < fetchedSql.length; i++) {
  let av = new cmodule.AliasValue();
  av.setValueString(fetchedSql[i]);
  outputAliases.push(av);
}

this.insertValueInMultiMap('Multi.ExampleModuleJS.SqlSelectOtherDb', outputAliases);
```

20.2 AliasValue class

The AliasValue class is defined in V8 “cmodule” object. So, the AliasValue class needs to be called with “cmodule”. For example**:** cmodule.AliasValue

20.2.1 AliasValue constructor

- **Purpose:** create AliasValue object

20.2.2 setValueDouble(value), setValueInt(value), setValueString(value)

- **Purpose:** set AliasValue object with double value, integer value, or string value

20.2.3 valueToString(), valueToInt(), valueToDouble()

- **Purpose:** return the string value, integer value or double value

20.2.4 isValueNull()

- **Purpose:** check if the Alias value is uninitialized

20.2.5 markForDeletion()

- **Purpose:** delete database table row pointed by Alias value after execution of AF extension function

20.2.6 isMarkForDeletion()

- **Purpose:** check if the Alias value is for deletion

Note: See the helper functions for AF for an example usage of AliasValue class.

20.3 AliasView class for JavaScript plugins

The AliasView class is defined in V8 “cmodule” object. So, the AliasView class needs to be called with “cmodule”. For example**:** cmodule.AliasView

20.3.1 AliasView(aliasView, aliasInput, ruleInput, userInformation, dataPoolLoader) constructor

- **Purpose:** create AliasView instance that evaluates alias view

20.3.2 Example

- This example creates AliasView instance and evaluates “CrunchView” alias view with input “Alias.Crunch.Contract” alias having value of “SAMPLEVUL”. The evaluate method stores the evaluated data to the $evaluatedData->returnData argument.

```
const aliasViews = JSON.stringify([ 'CrunchView' ]);
const aliasInput = JSON.stringify({ 'Alias.Crunch.Contract': 'SAMPLEVUL' });
const av = new cmodule.AliasView(aliasViews, aliasInput, funcParam, funcParam, this.poolLoader);

const evaluatedData = JSON.parse(av.evaluate());
if (evaluatedData["result"]) {
    const returnData = evaluatedData["returnData"];
```

20.4 JSON request helper functions for JavaScript plugins

These helper functions parse the JSON request and provide convenient way to retrieve and update the values of aliases. This is are normally use for plugins that going to be assigned as pre-extension functions.

These functions are defined in V8 “cmodule” object. So, the functions needs to be called with “cmodule”. For example**:** cmodule.getAliasValue

20.4.1 getAliasValue(aliasName)

- **Purpose:** retrieve the string value of an alias
- **Example:** This example retrieve the value of “Alias.EmailSendTests.language” alias from JSON request.

```
var pluginLanguage = cmodule.getAliasValue("Alias.EmailSendTests.language");
```

20.4.2 setAliasValue(aliasName, value)

- **Purpose:** modify the value of an alias
- **Example:** This example modifies “Alias.Portfolio.Description” alias of the JSON request to “new description”.

```
cmodule.setAliasValue("Alias.Portfolio.Description", "new description");
```

20.4.3 getAliasArraySize()

- **Purpose:** returns the size of alias array
- **Example:** This example stores the size of array to variable $arraySize

```
var arraySize = cmodule.getAliasArraySize();
```

20.4.4 getAliasArrayValue(index, aliasName)

- **Purpose:** retrieve the string value of array by index
- **Example:** This example retrieves 4th string value of “Array.PortfolioFundsTEST.FundNo” alias from JSON request

```
var getAliasArrayValue = cmodule.getAliasArrayValue(3, "Array.PortfolioFundsTEST.FundNo");
```

20.4.5 setAliasArrayValue(index, aliasName, value)

- **Purpose:** modify the value of an alias from an array index
- **Example:** This example replaces the 4th value of "Array.PortfolioFundsTEST.FundNo" with "5".

```
cmodule.setAliasArrayValue(3, "Array.PortfolioFundsTEST.FundNo", "5");
```

20.4.6 getAFValues()

- **Purpose:** return aliases (excludes alias array) from values of aliases of JSON request

20.4.7 getAFArrayValues()

- **Purpose:** return entire alias arrays (excludes non-array aliases) of JSON request

20.5 Message helper functions for JavaScript plugins

These functions are defined in V8 "cmodule" object. So, the functions needs to be called with "cmodule". For example**:** cmodule.isConfirmed

20.5.1 isConfirmed(messageCode)

- **Purpose:** Check if the JSON request has confirmed message_code. This helper function is use with plugin_helpers.INFO and plugin_helpers.WARNING.
- Note: See the examples of plugin_helpers.INFO and plugin_helpers.WARNING for usage of IsConfirmed function.

20.5.2 info(messageCode, message)

- **Purpose:** JSON response with info message
- **Example:** This example sends back a message response to client with an id of 1.

```
if(!cmodule.isConfirmed(1)) {
    return cmodule.info(1, "Starting a task");
}
```

- Client output:

- Note: The client will resend the request after it has acknowledge the info message. So, the plugin function must check the messageCode to skip the info message.

20.5.3 warning(messageCode, message)

- **Purpose:** JSON response with warning message
- **Example:** This example sends back a warning message response to client with an id of 2.

```
if(!cmodule.isConfirmed(2)) {
    return cmodule.warning(2, "Test WARNING from JS");
}
```

- Note: The client will resend the request after it has acknowledge the warning message. So, the plugin function must check the messageCode to skip the warning message.

20.5.4 error(messageCode, message)

- **Purpose:** JSON response with error message
- **Example:** This example sends back an error message response to client.

```
return cmodule.error(3, "Test ERROR from JS");
```

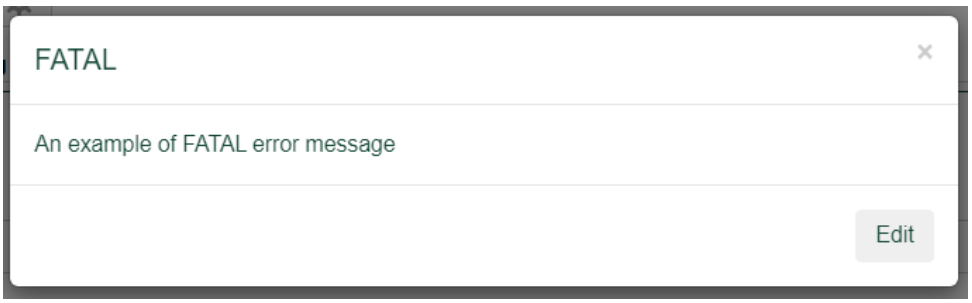
- Note: The client doesn't resend the request after it has acknowledge the error message.

20.5.5 fatal(messageCode, message)

- **Purpose:** JSON response with fatal message
- **Example:** This example sends back an fatal message response to client.

```
return cmodule.fatal(4, "An example of FATAL error message");
```

- Client output:



- Note: The client doesn't resend the request after it has acknowledge the fatal message.

20.6 Document management helper functions for JavaScript plugins

20.6.1 cmodule.copyDocumentToDevice(filePath, fileDescription, documentFamily, documentCategory)

- **Purpose:** Copies document to a specific device, the family and category are optional parameters, if they are not specified the document is copied according to the parameters in the JsonKeeper.
- **Return:** Returns the file container ID if successful, otherwise an empty string is returned.
- **Note:** The example for this function is provided together with the “addDocumentToReturnList” function example.

20.6.2 cmodule.addDocumentToReturnList(documentID)

- **Purpose:** Adds the file container ID to the JsonKeeper and later in the response of the extension function.
- **Return:** Returns “true” if successfully executed, otherwise “false” is returned.
- **Example:**

```
var isAdded = false;
var fileID = '';
var filePath = 'C:\\DocumentManagementTest.txt';
var fileDescription = 'description';

var documentFamily = 'test';
var documentCategory = 'test';

// copies the document to the device specified on the screen
fileID = cmodule.copyDocumentToDevice(filePath, fileDescription);
// adds the file container ID in the response
isAdded = cmodule.addDocumentToReturnList(fileID);
cmodule.print('File container ID: ' + fileID + '. Added to return list: ' + isAdded + '.');

// copies the document to the device specified in the parameters
fileID = cmodule.copyDocumentToDevice(filePath, fileDescription, documentFamily, documentCategory);
// adds the file container ID in the response
isAdded = cmodule.addDocumentToReturnList(fileID);
cmodule.print('File container ID: ' + fileID + '. Added to return list: ' + isAdded + '.');

return system.prepareResponse(true, 'OK');
```

20.6.3 cmodule.getDocumentFromDevice(fileContainerID, filePath)

- **Purpose:** Loads the document with the specified file container ID in the specified path.
- **Return:** Returns “true” if successfully executed, otherwise “false” is returned.
- **Example:**

```
var isLoaded = false;
var fileID = '1111';
var filePath = 'D:\\Users\\spoposki\\Desktop\\testFile.jpg';

// loads the document with the specified file container ID in the specified path
isLoaded = cmodule.getDocumentFromDevice(fileID, filePath);
cmodule.print('File container ID: ' + fileID + '. Is loaded: ' + isLoaded + '. Location: ' + filePath);

return system.prepareResponse(true, 'OK');
```

20.6.4 cmodule.deleteDocumentFromDevice(fileContainerID, performHardDelete)

- **Purpose:** Sets the file container “Active” flag to “false” or deletes the file container completely, it depends on the value of the “performHardDelete” parameter.
- **Returns:** Returns “true” if successfully executed, otherwise “false” is returned.
- **Example:**

```
var isDeleted = false;
var fileID = '1111';

// performs a soft delete on the specified file container
isDeleted = cmodule.deleteDocumentFromDevice(fileID, false);
cmodule.print('File container ID: ' + fileID + '. Is soft deleted: ' + isDeleted);

// performs a complete delete on the specified file container
isDeleted = cmodule.deleteDocumentFromDevice(fileID, true);
cmodule.print('File container ID: ' + fileID + '. Is completely deleted: ' + isDeleted);

return system.prepareResponse(true, 'OK');
```

20.6.5 cmodule.searchForDocument(documentFamily, documentCategory, kp1, kp2, kp3, kp4, kp5, description)

- **Purpose:** Searches for file containers using the function parameters as search parameters. If a function parameter is an empty string that field will not be included in the search.
- **Return:** Returns serialized JSON array containing information about the file containers that match the search parameters.
- **Example:**

```
var fileRootPath = 'D:\\Users\\spoposki\\Desktop\\';
var description = 'test1';

// searches for the documents that have the value "test" in description
var files = cmodule.searchForDocument('', '', '', '', '', '', '', description);
cmodule.print('Files: ' + files);
// parses the response and iterates through the array
var filesArray = JSON.parse(files);
filesArray.forEach(function (file, index) {
  var isLoaded = false;
  var filePath = fileRootPath + file['container_file_name']
  // loads the document to disk
  isLoaded = cmodule.getDocumentFromDevice(file['id'].toString(), filePath);
  cmodule.print('File container ID: ' + file['id'] + '. Is loaded: ' + isLoaded + '. Location: ' + filePath);
});

return system.prepareResponse(true, 'OK');
```

20.6.6 cmodule.setDocumentKeyParts(fileContainerID, kp1, kp2, kp3, kp4, kp5)

- **Purpose:** Sets the file container key parts.
- **Return:** Returns “true” if successfully executed, otherwise “false” is returned.
- **Example:**

```
var isSet = false;

var fileID = '1111';
var kp1 = 'test_kp1';
var kp2 = 'test_kp2';
var kp3 = 'test_kp3';
var kp4 = 'test_kp4';
var kp5 = 'test_kp5';

// sets the key parts
isSet = cmodule.setDocumentKeyParts(fileID, kp1, kp2, kp3, kp4, kp5);
cmodule.print('Key parts set: ' + isSet);

return system.prepareResponse(true, 'OK');
```

20.7 Database helper functions for JavaScript plugins and JavaScript AF extension functions

These functions are defined in V8 “cmodule” object. So, the functions needs to be called with “cmodule”. For example**:** cmodule.executeSQL

20.7.1 DBCursor

- **Purpose:** store the query results of execute_sql; and able to fetch row sequentially from that query results
- **Methods:**
  - size() returns number of rows
  - setPosition(position) set the position of cursor in query results
  - getPosition() return the position of cursor in query results
  - fetchOne() returns row (in list of columns); and next fetch_one() call returns next row. Note that the column names are in alphabetic order.
  - columnNames() returns the column names of query results. The column names are in alphabetic order.
- **Note:** See the example of executeSQL function for usage

20.7.2 executeSQL(sqlQuery, dbConnector)

- **Purpose:** query the database by running SQL query to Codiac database or to other database (with DB connection information)
- Note: dbConnector parameter is optional and default to empty which means to connect to Codiac database
- **Example:** This example run SQL query to a database (with DB connection information) and dump the first row result.

```
var dbConnector = new cmodule.DbConnector("Postgres", "[redacted]", "5432", "[redacted]", "postgres", "[redacted]");
var sqlResult = cmodule.executeSQL('SELECT * FROM \"PolicyAgent\"',
                                   dbConnector);

print("DB Cursor size: ", sqlResult.size());

var columnNames = sqlResult.columnNames();
print("DB column names:", columnNames);

var fetchedSql = sqlResult.fetchOne();
print("DB fetch one:", fetchedSql);
print("GetPosition after fetching the result: ", sqlResult.getPosition());
```

20.8 Other CodiacSDK helper functions for JavaScript plugins and JavaScript AF extension functions

These functions are defined in V8 “cmodule” object. So, the functions needs to be called with “cmodule”. For example**:** cmodule.sendEmail

20.8.1 sendEmail(to, subject, body)

- **Purpose:** send an email to recipient with subject and message
- **Example:** This AF extension function example sends an email with recipient from “Alias.EmailSendTests.recipient” alias, subject from “Alias.EmailSendTests.subject” alias and message from “Alias.EmailSendTests.body” alias.

```
let avRecipient = this.getValueFromMap("Alias.EmailSendTests.recipient");
const recipient = avRecipient.valueToString();
let avSubject = this.getValueFromMap("Alias.EmailSendTests.subject");
const subject = avSubject.valueToString();
let avBody = this.getValueFromMap("Alias.EmailSendTests.body");
const body = avBody.valueToString();
cmodule.sendEmail(recipient, subject, body);
```

20.8.2 sendEmail(to, subject, body, attachments)

- **Purpose:** Sends an email to recipient with subject, message and attachments. The attachment parameter is a delimited string of file container IDs.
- **Return:** A serialized JSON with details.
- **Example:**

```
var to = 'slavkopoposki@gmail.com';
var subject = 'SUBJECT';
var body = 'BODY';

var files = '1178;1179;1180';

// sends an email with attachments, the file that correspond to the id will the attached to the email
cmodule.sendEmail(to, subject, body, files);

return system.prepareResponse(true, 'OK');
```

20.8.3 sendSms(to, body)

- **Purpose:** send SMS message to recipient with message
- **Example:** This example send SMS message to 5556789090 (not a real phone number). Only US numbers are accepted.

```
cmodule.sendSms("5556789090","This is a test message.");
```

20.8.4 generateReport(reportName, inlineJson, isSaveToXml)

- **Purpose:** generate report PDF file from report template (client report builder)
- Note: The isSaveToXml default to false; if set to true, the report are save to XML file and is not generated to PDF.
- **Example:** This example generate report from report template “Test CrunchF Report” with JSON data from an alias view data.

```
const aliasViewData = jsonParams['alias_view_data'];
let inlineJson = '';
if (Array.isArray(aliasViewData) && aliasViewData.length > 0) {
  inlineJson = JSON.stringify(aliasViewData[0]);
} else {
  inlineJson = JSON.stringify(aliasViewData);
}

return cmodule.generateReport('Test Crunch Report', inlineJson);
```

- **Example:** Another example that creates JSON data from aliases.

```
let jsonInput = {};

jsonInput['Array.CrunchF'] = [];
for (let i = 0; i < contracts.length; i++)
{
  const item = {
    'Array.CrunchF.Contract': contracts[i].valueToString(),
    'Array.CrunchF.Description': descriptions[i].valueToString(),
    'Array.CrunchF.FundNo': fundNos[i].valueToString(),
    'Array.CrunchF.Amt': amounts[i].valueToString()
  };
  jsonInput['Array.CrunchF'].push(item);
}

// generate report
cmodule.generateReport('Test CrunchF Report', JSON.stringify(jsonInput));
```

20.8.5 generateReport(reportName, inlineData, delimiter, isSaveToXml)

- **Purpose:** generate report PDF file from report template (client report builder)

- Note: The isSaveToXml default to false; if set to true, the report are save to XML file and is not generated to PDF.
- **Example:** This example generate report from report template “Test CrunchF Report” with values from vector of delimited values. The delimiter used '|'.

```
const inlineData = [
  'Contract|FundNo|Description|Amount',
  'Contract A|CA1|Test contract A - 1|123.45',
  'Contract A|CA2|Test contract A - 2|3.45',
  'Contract B|CB1|Test contract B - 1|645',
  'Contract C|CC1|Test contract C - 1|7.89'
]

return cmodule.generateReport('Test CrunchF Report - inline data', inlineData, '|');
```

20.8.6 executeJson(jsonRequest)

- **Purpose:** call Codiac service by executing JSON request
- **Example:** This example calls JobsScheduler::GetJobsScheduleList to get list of jobs. A JSON request is created with func_name parameter and lib_name parameter.

```
let jsonRequest = {
  'requestbody': {
    'func_name': 'GetJobScheduleList',
    'lib_name': 'CodiacSDK.zJobsScheduler'
  }
};

const jsonResponse = cmodule.executeJson(JSON.stringify(jsonRequest));
```

- Note: The user\session parameters are filled up at service process.

20.9 SQL query with JavaScript native database modules

Codiac SDK provides database helper functions that can access Codiac database and other databases with connection string. In addition, the user can directly install npm database package and import to JavaScript to query database.

The DirectSql plugins contain example source code on how to query the databases with native database modules

- **PostgreSQL**

  The example source codes uses node-postgres module to query PostgreSQL database. To install this module, go to service root directory and run:

  npm install pg

This will download and install the pg package to node_modules directory. Then, the JavaScript plugin\module can 'require' this package and call the query functions.

Below is the sequence for pg to query database:

1. Create Client object with database connection information
2. Call “connect” function asynchronously on Client object
3. Call “query” function asynchronously with SQL query on Client object
   - return query result object
4. Call “forEach” function to fetch query result
5. Call “end” function asynchronously to disconnect client

The PostgreSQL function returns a “promise” to handle the asynchronous database query. See the PostgreSQL function of DirectSqlExample.js or ExampleModule.js for full example.

See the https://node-postgres.com/ for full details on how to use node-postgres module.

- **MySQL**

  The example source codes uses mysql module to query MySQL database. To install this module, go to service root directory and run:

  npm install mysql

  This will download and install the mysql package to node_modules directory. Then, the JavaScript plugin\module can 'require' this package and call the query functions.

  Below is the sequence for mysql to query database:

  1. Create connection object with database connection information
  2. Call “connect” function on connection object
  3. As a callback of “connect” function, call “query” function with SQL query on connection object
  4. As a callback of “query” function, call another “query” function (without query result)
  5. As a callback of “query” function, call “forEach” function to fetch query result

  The MYSQL function returns a “promise” to handle the asynchronous database query. See the MYSQL function of DirectSqlExample.js or ExampleModule.js for full example.

  See the https://www.npmjs.com/package/mysql for full details on how to use mysql module.

- **MS SQL Server**

  The example source codes uses mssql module to query MS SQL Server database. To install this module, go to service root directory and run:

  npm install mssql

  This will download and install the pg package to node_modules directory. Then, the JavaScript plugin\module can 'require' this package and call the query functions.

Below is the sequence for mssql to query database:

1. Create configuration object with database connection information
2. Call “connect” function asynchronously with configuration object to connect to database
3. Call “query” function asynchronously with SQL query on connect object
    - return query result object
4. Call “forEach” function to fetch query result

The MSSQL function returns a “promise” to handle the asynchronous database query. See the MSSQL function of DirectSqlExample.js or ExampleModule.js for full example

See the https://www.npmjs.com/package/mssql for full details on how to use mssql module

- **Oracle**

The example source codes uses oracledb module to query Oracle database. To install this module, go to service root directory and run:

npm install oracledb

This will download and install the oracledb package to node_modules directory. Then, the JavaScript plugin\module can 'require' this package and call the query functions.

Below is the sequence for oracledb to query database:

1. Call “getConnection” function with database connection information to connect to Oracle SQL database
2. Call “execute” function on connection object with SQL query
    - return query result object
3. Call “forEach” function to fetch query result

The OracleSQL function returns a “promise” to handle the asynchronous database query. See the OracleSQL function of DirectSqlExample.js or ExampleModule.js for full example.

See the https://www.npmjs.com/package/oracledb for full details on how to use oracledb module.

21 JavaScript Debugging with Visual Studio Code

Before starting the debugger, create .vscode\launch.json file for JavaScript debugging as shown below. Port 9229 can be use for plugin and module debugging.

```
{
  // Use IntelliSense to learn about possible attributes.
  // Hover to view descriptions of existing attributes.
  // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
  "version": "0.2.0",
  "configurations": [
    {
      "type": "node",
      "request": "attach",
      "name": "Attach",
      "port": 9229,
      "localRoot": "${workspaceFolder}",
      "remoteRoot": "."
    }
  ]
}
```

While the service is running, starts VS Code debugger:

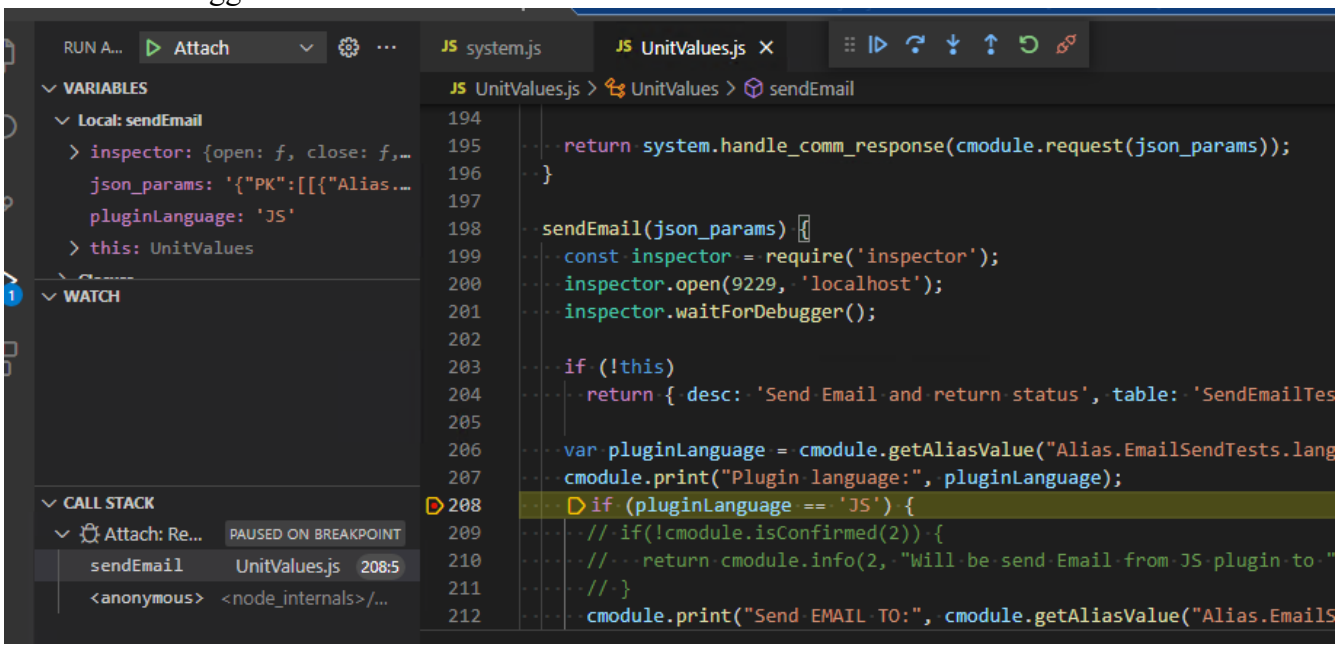
1. Launch VS Code in administrator account
2. Open the JavaScript directory to be debugged
   - AF extension function: x64\<*Debug*|*Release*>\Modules\js
   - plugins: x64\<*Debug*|*Release*>\plugins\js
3. Open the JavaScript script to be debugged
4. Insert nodejs inspector (red box) in JavaScript script to be debugged. The port must match with the port of launch.json file.

```
ExternalDb () {
  const inspector = require('inspector');
  inspector.open(9229, 'localhost');
  inspector.waitForDebugger();

  const fs = require('fs');

  return extModule.SaveCrunchfToOtherDb('crunchf_example.db', this.getCrunchF())
    .then(() => extModule.GetCrunchfFromOtherDb('crunchf_example.db', '10SP'))
    .then(function(crunchfList) {
```

5. After the inspector, add breakpoints by clicking on left side of line number.
6. Execute the JavaScript plugin or JavaScript AF extension function.
7. When the inspector starts listening, the console output will have “Debugger listening ..” as shown below:

```
Debugger listening on ws://localhost:9229/d2817c4c-9cad-4cc9-aba5-0e6adb9f5df5
Debugger listening on ws://localhost:9229/d2817c4c-9cad-4cc9-aba5-0e6adb9f5df5
For help, see: https://nodejs.org/en/docs/inspector
```

8. Then, click the debug icon on left side bar
9. Select “Attach”
10. Click the green play button (or F5) to start debugging. At this point, VS Code is attached to the debugger.



11. You can step through the codes by pressing the F10 (step next), F11 (step into), and F5 (continue) keys or detach (orange chain icon). You can add watches or see local watches.

22 Remote Debugging JavaScript plugin and AF extension functions

In the JavaScript language, remote debugging is almost identical to local debugging.
In the Remote Debugger in the **Attach to Process** step, select the IP address from the **Connection target** drop-down list. The IP address should belong to the remote machine.
Then, perform the debugging steps as for the local debugging described above in the Debugging section.

23 C# Alias Framework extension module

The Alias Framework (AF) service can evaluates custom generated aliases or multi generated aliases. This is done by going thru the alias name structure:

AliasType.ModuleName.FunctionName

Depending on the AliasModuleType, the custom alias can be C++ function, Python function, JavaScript function, PHP function, or C# function.

For example, the "Custom.ExampleModuleCS.CalculateAverage" tells us that the:

- alias type is custom generated alias,
- if the AliasModuleType is C#, then C# class name is ExampleModuleCS and method name is CalculateAverage.

Then, AF spawns ScriptExecutorNet.exe with class name and method name information. The ScriptExecutorNet looks in `<service directory>\Modules\net` directory for DLL file that has a ExampleModuleCS class deriving from IAfModule interface. Then, ScriptExecutorNet executes the CalculateAverage method.

To summarize the requirements of C# AF extension module,

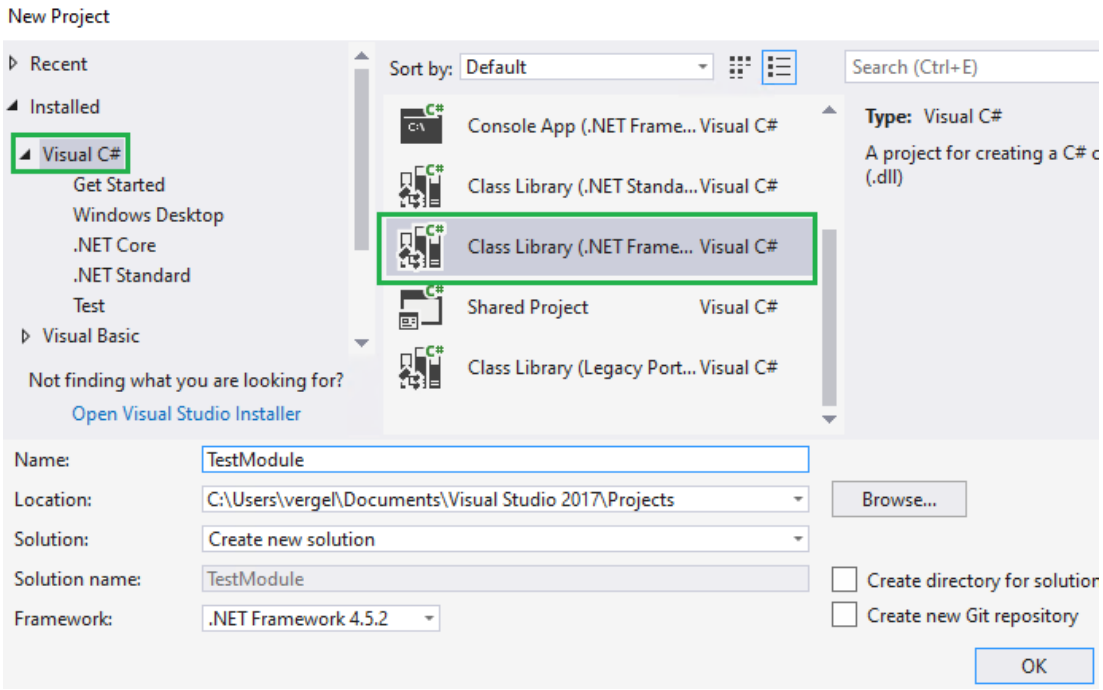
- a public class which derived from IAfModule interface
- public methods with no parameter
- the output DLL file is place in `<service directory>\Modules\net` directory

23.1 Required tools

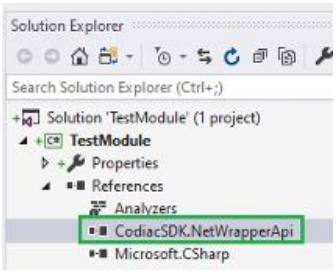
- Visual Studio 2017 or later
  - .NET desktop development for C#
  - .Net Core
    - For the moment, the example C# modules uses Net Framework 4.5.2

23.2 How to create C# AF extension function?

1. Create a C# class library project with .NET framework in Visual Studio (see example below)



2. Add CodiacSDK.NetWrapperApi.dll from <*Codiac root directory*> to References. This library provides Codiac service APIs.



3. Add other .Net DLL file to References as needed. For example**:**

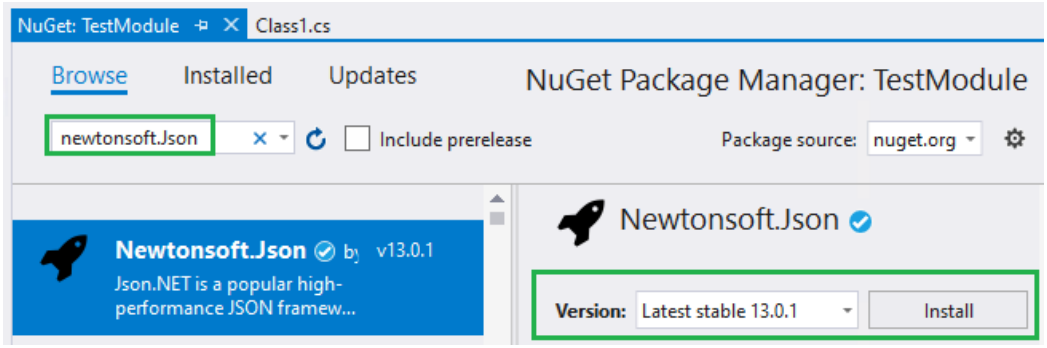
   a) An external DLL file from a different directory that is going to be called by the project.

   - If this is the case, add the library directory path to probing of ScriptExecutorNet.exe.config (in service root directory).

ScriptExecutorNet.exe.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.2" />
    </startup>

  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">

      <!-- use probing to specify which path to load assemblies -->
      <probing privatePath="path\to\other_modules" />

    </assemblyBinding>
  </runtime>

</configuration>
```

4. Add Newtonsoft.Json package to project in NuGet Package Manager



5. Add other packages as needed. For example:
   a) MySql.Data package by Oracle to access MySQL database
   b) Npsql packaage to access PostgreSQL database
   c) Oracle.ManagedDataAccess package by Oracle to access Oracle database

6. Create a public class that derived from AfModule class which is defined in CodiacSDK.NetWrapperAPI assembly. Below is an example of a class (named TestModuleCS) derived from AfModule.

```
using CodiacSDK.NetWrapperApi;

namespace TestModuleCS
{
    public class TestModuleCS : AfModule
    {

    }
}
```

7. Create methods that are going to be executed by AF. Below is an example of two empty methods. See examples in Helper functions on how to make tasks or get\set the values of

aliases.

```
public class TestModuleCS : AfModule
{
    public void Calculation()
    {

    }

    public void Task()
    {

    }
}
```

8. Before the functions can exit, the alias of extension function must be added to evaluated aliases.
   - For Multi Generated alias, set the alias to a list of AliasValue objects.
   - For Custom Generated alias, set the alias to an AliasValue object.

   See below for an example:

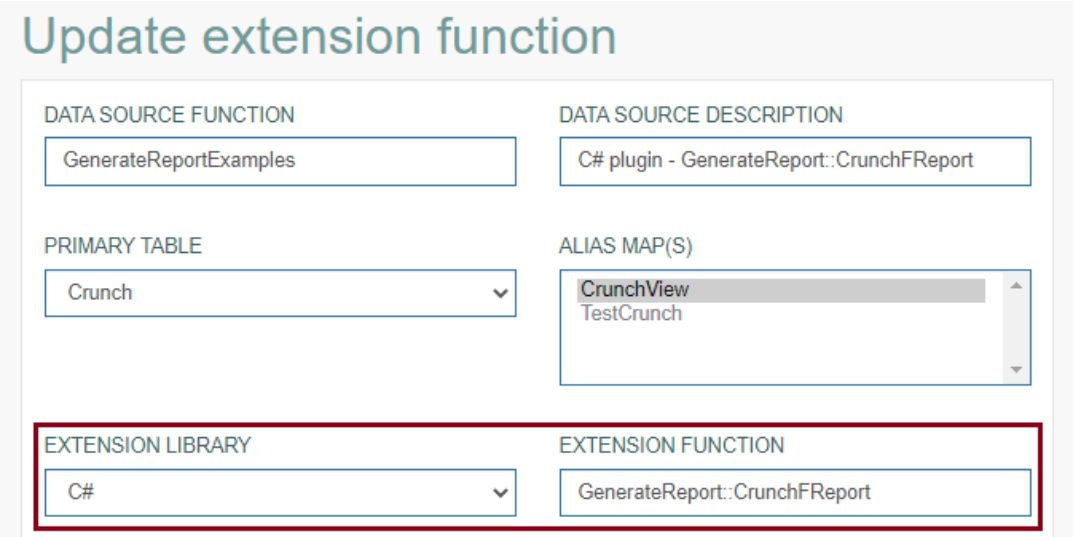
```
public void Calculation()
{
    // some calculations here

    AliasValue av1 = new AliasValue();
    av1.SetValue(1.2345);

    AliasValue av2 = new AliasValue();
    av1.SetValue(3.1416);

    InsertInMultiMap("Multi.TestModuleCS.Calculation",
        new List<AliasValue>() { av1, av2 });
}

public void Task()
{
    // some tasks here

    AliasValue av = new AliasValue();
    av.SetValue("Completed a task");

    InsertInMap("Custom.TestModuleCS.Task", av);
}
```

9. Build the project and copy the output DLL file and referenced DLL files to *<Codiac service root directory>*\Modules\net directory.
10. Finally, add the Custom\Multi Generated alias of your AF extension function in Codiac builder.
    a) Add your Custom\Multi Generated alias to Alias setup
    b) Configure your screen to include the your Custom\Multi Generated alias
    c) Execute screen in Codiac render to test your AF extension function

24 C# extension functions (plugins)

The extension functions are configured in PHP client. Screenshot below is an example setup which specifies the class and the method to execute. This is different from C++ setup which specifies the user module name.

The CodiacSDK.NetWrapper service handles the loading and execution of C# class\method. So, this service is called instead of user module.



Let say, we have “NetPlugin.cs” source code having a “APlugin” class with “AMethod” method. To set it up as an C# extension function,

1. In PHP client, go to **System → Extension function → Create**
2. Select “C#” in **EXTENSION LIBRARY**
3. Put class name and function name in EXTENSION FUNCTION with double colon in between (**Example:** “APlugin::AMethod”)

The setup will look like below.

Note that the source code name (“NetPlugin.cs”) or assembly name (“NetPlugin.dll”) is not use in the setup.

24.1 Required tools

- Visual Studio 2017 or later
  - .NET desktop development for C#
  - .Net Core

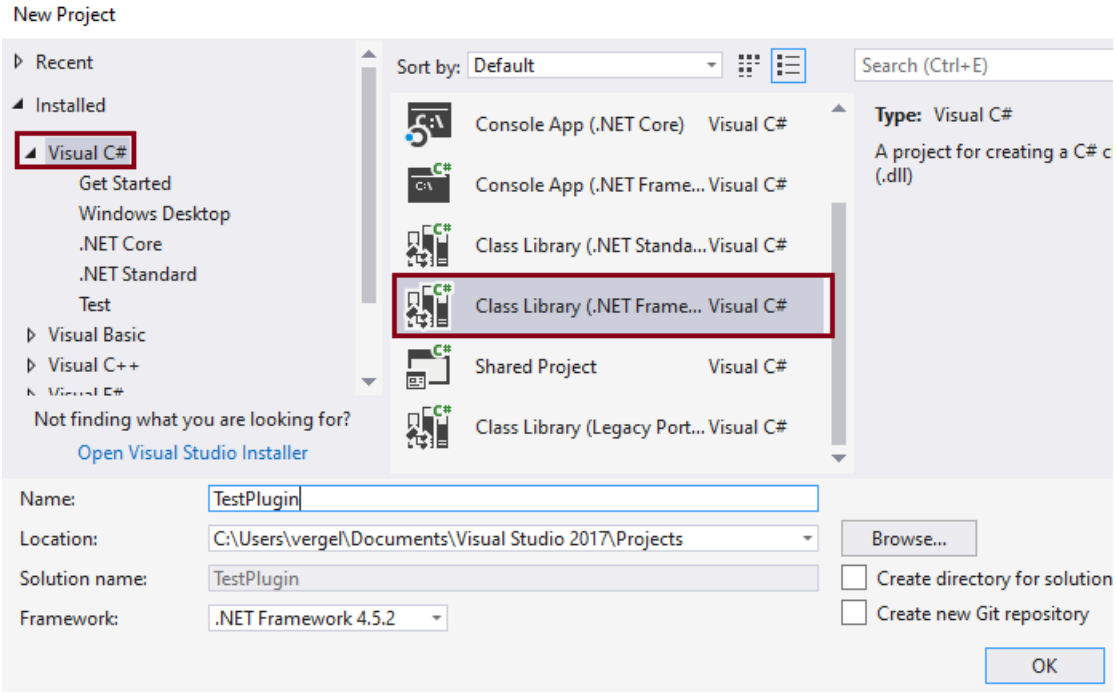
- For the moment, the example C# modules uses Net Framework 4.5.2

24.2 Structure of C# plugin

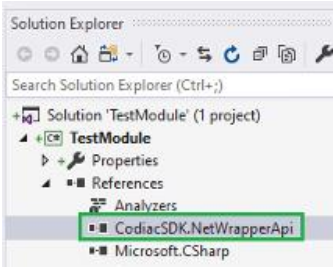
- The plugin class derives from PluginModule class which is defined in CodiacSDK.NetWrapperApi assembly.
- The PluginModule class contains the implementation for sdk_module_info method and configure method. These two methods are called first before the plugin function.

24.3 How to create a C# plugin?

1. Create a C# class library project with .NET framework in Visual Studio (see example below)



2. Add CodiacSDK.NetWrapperApi.dll from <*Codiac root directory*> to References. This library provides Codiac service APIs.



3. Add other .Net DLL file to References as needed. For example**:**

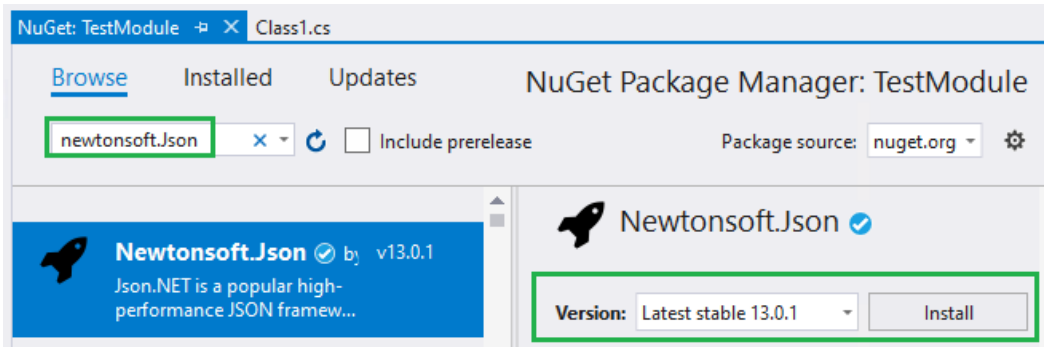
   a) An external DLL file from a different directory that is going to be called by the project.

   - If this is the case, add the library directory path to probing of ScriptExecutorNet.exe.config (in service root directory).

ScriptExecutorNet.exe.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.2" />
    </startup>

  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">

      <!-- use probing to specify which path to load assemblies -->
      <probing privatePath="path\to\other_modules" />

    </assemblyBinding>
  </runtime>

</configuration>
```

4. Add Newtonsoft.Json package to project in NuGet Package Manager



5. Add other packages as needed. For example:
   a) MySql.Data package by Oracle to access MySQL database
   b) Npsql packaage to access PostgreSQL database
   c) Oracle.ManagedDataAccess package by Oracle to access Oracle database
6. Create a public class that derived from PluginModule class which is defined in CodiacSDK.NetWrapperAPI assembly. Below is an example of a class (named TestPluginCS) derived from PluginModule.

```
using CodiacSDK.NetWrapperApi;

namespace TestPluginCS
{
    public class TestPluginCS : PluginModule
    {

    }
}
```

7. Create method that is going to be executed by CodiacSDK.NetWrapper. Below is an example of an empty method. See the example of C# helper functions on how to add

implementations to plugin functions.

```
public class TestPluginCS : PluginModule
{
    public void Task(JObject jsonParams)
    {

    }
}
```

8. Build the project and copy the output DLL file and referenced DLL files to *<Codiac service root directory>*\plugins\net directory.
9. Finally, add your plugin functions in Codiac builder.
   a) Add your plugin functions to extension function setup
   b) Configure your screen to include the extension functions
   c) Execute screen in Codiac render to test your plugin functions

25 C# helper functions

These helper functions will allow you to access to Alias Framework, query database, send email, send SMS message, and execute Codiac services.

Note that some functions are only available for AF extension functions, some are only for plugins and other functions are for both.

25.1 AF helper functions for AF extension functions

25.1.1 GetValueFromMap(*aliasName*)

- **Purpose:** return AliasValue object of an alias, or null if not found
- **Example:** This example get the AliasValue object of "Alias.EmailSendTests.language". With the AliasValue object, the value can be retrieve using string Value property.

```
AliasValue language = GetValueFromMap("Alias.EmailSendTests.language");
string lang = language?.Value;
```

25.1.2 GetValueFromMultiMap(aliasName)

- **Purpose:** return array of AliasValue objects of multi-value alias, or return null if not found
- **Example:** This example get the list of AliasValue objects of "Array.CrunchF.Contract" and other array aliases. Then, iterate on that list of AliasValue objects by index.

```
List<AliasValue> contracts = GetValueFromMultiMap("Array.CrunchF.Contract");

for (int i = 0; i < contracts.Count; i++)
{
    JObject crunchF = new JObject
        {
            { "Array.CrunchF.Contract", contracts[i].Value }
```

25.1.3 ReplaceOrInsertValueInMap(aliasName, aliasValue)

- **Purpose:** insert alias (with AliasValue object) in evaluated aliases
- Example:This example creates an AliasValue object and add it in evaluated aliases.

```
AliasValue av = new AliasValue() { Value = "sent email from C# module" };
ReplaceOrInsertValueInMap("Custom.ExampleModuleCS.Email", av);
```

25.1.4 ReplaceOrInsertValueInMultiMap(aliasName, aliasValues)

- **Purpose:** insert multi-value Alias (with vector of AliasValue object) in evaluated aliases
- Example:This example creates list of AliasValue objects and add it in evaluated aliases.

```
IEnumerable<AliasValue> outData = sqlResult.FetchOne().Select(item =>
    new AliasValue() { Value = item });
ReplaceOrInsertValueInMultiMap("Multi.ExampleModuleCS.SqlSelectOtherDb", outData.ToList());
```

25.2 AliasValue class

25.2.1 AliasValue constructor

- **Purpose:** create AliasValue object

25.2.2 Value property

- **Purpose:** set or get string value of AliasValue object

25.2.3 SetValue(value)

- **Purpose:** set value of AliasValue object with double value, integer value, or string value

25.2.4 ValueToString(), ValueToInt(), ValueToDouble()

- **Purpose:** return the string value, integer value or double value

25.2.5 IsValueNull()

- **Purpose:** check if the Alias value is uninitialized

25.2.6 MarkForDeletion()

- **Purpose:** delete database table row pointed by Alias value after execution of AF extension function

25.2.7 IsMarkForDeletion()

- **Purpose:** check if the Alias value is for deletion

Note: See the helper functions for AF for an example usage of AliasValue class.

25.3 AliasView class for C# plugins

25.3.1 AliasView(view, aliasInput, ruleInput, userInfo, dataPoolLoader) constructor

- **Purpose:** create AliasView instance that evaluates alias view

25.3.2 Example

- This example creates AliasView instance and evaluates “CrunchView” alias view with input “Alias.Crunch.Contract” alias having value of “SAMPLEVUL”. The evaluate method stores the evaluated data to the evaluatedData[“returnData“] object.

```
string[] aliasViews = new string[] { "CrunchView" };
JObject aliasInput = new JObject
{
    {  "Alias.Crunch.Contract", "SAMPLEVUL" }
};

var av = new CodiacSDKApi.AliasView(aliasViews, aliasInput, jsonParams, jsonParams, DataLoader);
var evaluatedData = av.Evaluate();

bool status = evaluatedData.Value<bool>("result");
JToken returnData = evaluatedData["returnData"];
```

25.4 JSON request helper functions for C# plugins

These helper functions parse the JSON request and provide convenient way to retrieve and update the values of aliases. This is are normally use for plugins that going to be assigned as pre-extension functions.

25.4.1 AliasValues[aliasName] property

- **Purpose:** get or set the string value of an alias
- **Example:** This example retrieve the value of "Alias.Portfolio.Description" alias from JSON request.

```
var val = AliasValues["Alias.Portfolio.Description"];
```

- **Example:** This example modifies "Alias.Portfolio.Description" alias of the JSON request to "new description".

```
AliasValues["Alias.Portfolio.Description"] = "new description";
```

25.4.2 AliasValues.Size property

- **Purpose:** returns the size of alias array
- **Example:** This example stores the size of array to variable arraySize

```
int arraySize = AliasValues.Size;
```

25.4.3 AliasValues[index, aliasName] property

- **Purpose:** get or set the string value of an array by an index
- **Example:** This example retrieves 4th string value of "Array.PortfolioFundsTEST.FundNo" alias from JSON request

```
var arrVal = AliasValues[3, "Array.PortfolioFundsTEST.FundNo"];
```

- **Example:** This example replaces the 4th value of "Array.PortfolioFundsTEST.FundNo" with "5".

```
AliasValues[3, "Array.PortfolioFundsTEST.FundNo"] = "5";
```

25.4.4 AliasValues.getAfValues()

- **Purpose:** return aliases (excludes alias array) from values of aliases of JSON request

25.4.5 AliasValues.GetAfArrayValues()

- **Purpose:** return entire alias arrays (excludes non-array aliases) of JSON request

25.5 Message helper functions for C# plugins

25.5.1 CodiacSDKApi.IsConfirmed(messageCode)

- **Purpose:** Check if the JSON request has confirmed message_code. This helper function is use with CodiacSDKApi.Info and CodiacSDKApi.Warning.
- **Note:** See the examples of CodiacSDKApi.Info and CodiacSDKApi.Warning for usage of CodiacSDKApi.IsConfirmed function.

25.5.2 CodiacSDKApi.Info(messageCode, message)

- **Purpose:** JSON response with info message
- **Example:** This example sends back a info message response to client with an id of 1. After client confirmation, the message code is included in the next request.

```
if (!CodiacSDKApi.IsConfirmed(1))
{
    return CodiacSDKApi.Info(1, "Starting a task");
}
```

- **Note:** The client will resend the request after it has acknowledge the info message. So, the plugin function must check the messageCode to skip the info message.
- **Client output:**

25.5.3 CodiacSDKApi.Warning(messageCode, message)

- **Purpose:** JSON response with warning message
- **Example:** This example sends back a warning message response to client with an id of 2. After client confirmation, the message code is included in the next request.

```
if (!CodiacSDKApi.IsConfirmed(2))
{
    return CodiacSDKApi.Warning(2, "Warning message");
}
```

- **Note:** The client will resend the request after it has acknowledge the warning message. So, the plugin function must check the messageCode to skip the warning message.

25.5.4 CodiacSDKApi.Error(messageCode, message)

- **Purpose:** JSON response with error message
- **Example:** This example sends back an error message response to client.

```
return CodiacSDKApi.Error(3, "Example error message");
```

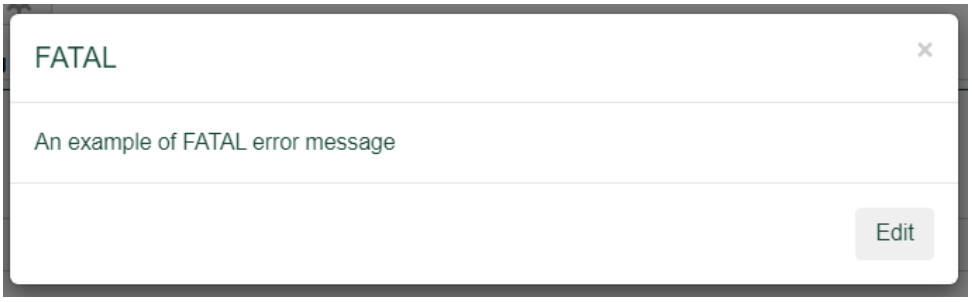
- **Note:** The client doesn't resend the request after it has acknowledge the error message.

25.5.5 CodiacSDKApi.Fatal(messageCode, message)

- **Purpose:** JSON response with fatal message
- **Example:** This example sends back a fatal message response to client.

```
return CodiacSDKApi.Fatal(4, "An example of FATAL error message");
```

- **Client output:**



- **Note:** The client doesn't resend the request after it has acknowledge the fatal message.

25.6 Document management helper functions for C# plugins

25.6.1 CodiacSDKApi.CopyDocumentToDevice(filePath, fileDescription, documentFamily, documentCategory)

- **Purpose:** Copies document to a specific device, the family and category are optional parameters, if they are not specified the document is copied according to the parameters in the JsonKeeper.
- **Return:** Returns the file container ID if successful, otherwise an empty string is returned.

- **Note:** The example for this function is provided together with the "AddDocumentToReturnList" function example.

25.6.2 CodiacSDKApi.AddDocumentToReturnList(documentID)

- **Purpose:** Adds the file container ID to the JsonKeeper and later in the response of the extension function.
- **Return:** Returns "true" if successfully executed, otherwise "false" is returned.
- **Example:**

```
bool isAdded = false;
string fileID = "";
string filePath = "C:\\DocumentManagementTest.txt";
string fileDescription = "description";

string documentFamily = "test";
string documentCategory = "test";

// copies the document to the device specified on the screen
fileID = CodiacSDKApi.CopyDocumentToDevice(filePath, fileDescription);
// adds the file container ID in the response
isAdded = CodiacSDKApi.AddDocumentToReturnList(fileID);
Log($"File container ID: {fileID}. Added to return list: {isAdded}.");

// copies the document to the device specified in the parameters
fileID = CodiacSDKApi.CopyDocumentToDevice(filePath, fileDescription, documentFamily, documentCategory);
// adds the file container ID in the response
isAdded = CodiacSDKApi.AddDocumentToReturnList(fileID);
Log($"File container ID: {fileID}. Added to return list: {isAdded}.");

return DataLoaderHelpers.PrepareResponse(status: true, message: "OK", data: null);
```

25.6.3 CodiacSDKApi.GetDocumentFromDevice(fileContainerID, filePath)

- **Purpose:** Loads the document with the specified file container ID in the specified path.
- **Return:** Returns "true" if successfully executed, otherwise "false" is returned.
- **Example:**

```
bool isLoaded = false;
string fileID = "1111";
string filePath = "D:\\Users\\spoposki\\Desktop\\testFile.jpg";

// loads the document with the specified file container ID in the specified path
isLoaded = CodiacSDKApi.GetDocumentFromDevice(fileID, filePath);
Log($"File container ID: {fileID}. Is isLoaded: {isLoaded}. Location: {filePath}");

return DataLoaderHelpers.PrepareResponse(status: true, message: "OK", data: null);
```

25.6.4 CodiacSDKApi.DeleteDocumentFromDevice(fileContainerID,

performHardDelete)

- **Purpose:** Sets the file container "Active" flag to "false" or deletes the file container completely, it depends on the value of the "performHardDelete" parameter.
- **Returns:** Returns "true" if successfully executed, otherwise "false" is returned.
- **Example:**

```
bool isDeleted = false;
string fileID = "1111";

// performs a soft delete on the specified file container
isDeleted = CodiacSDKApi.DeleteDocumentFromDevice(fileID, false);
Log($"File container ID: {fileID}. Is soft deleted: {isDeleted}");

// performs a complete delete on the specified file container
isDeleted = CodiacSDKApi.DeleteDocumentFromDevice(fileID, true);
Log($"File container ID: {fileID}. Is completely deleted: {isDeleted}");

return DataLoaderHelpers.PrepareResponse(status: true, message: "OK", data: null);
```

25.6.5 CodiacSDKApi.SearchForDocument(documentFamily, documentCategory, kp1, kp2, kp3, kp4, kp5, description)

- **Purpose:** Searches for file containers using the function parameters as search parameters. If a function parameter is an empty string that field will not be included in the search.
- **Return:** Returns serialized JSON array containing information about the file containers that match the search parameters.
- **Example:**

```
string fileRootPath = "D:\\Users\\spoposki\\Desktop\\";
string description = "test1";

// searches for the documents that have the value "test" in description
string files = CodiacSDKApi.SearchForDocument("", "", "", "", "", "", "", description);
Log($"Files: {files}");
// parses the response and iterates through the array
JArray filesArray = JArray.Parse(files);
foreach (JObject file in filesArray)
{
    bool isLoaded = false;
    string filePath = fileRootPath + file["container_file_name"].ToString();
    // loads the document to disk
    isLoaded = CodiacSDKApi.GetDocumentFromDevice(file["id"].ToString(), filePath);
    Log($"File container ID: {file["id"]}. Is isLoaded: {isLoaded}. Location: {filePath}");
}

return DataLoaderHelpers.PrepareResponse(status: true, message: "OK", data: null);
```

25.6.6 CodiacSDKApi.SetDocumentKeyParts(fileContainerID, kp1, kp2,

kp3, kp4, kp5)

- **Purpose:** Sets the file container key parts.
- **Return:** Returns “true” if successfully executed, otherwise “false” is returned.
- **Example:**

```
bool isSet = false;

string fileID = "1111";
string kp1 = "test_kp1";
string kp2 = "test_kp2";
string kp3 = "test_kp3";
string kp4 = "test_kp4";
string kp5 = "test_kp5";

// sets the key parts
isSet = CodiacSDKApi.SetDocumentKeyParts(fileID, kp1, kp2, kp3, kp4, kp5);
Log($"Key parts set : {isSet}");

return DataLoaderHelpers.PrepareResponse(status: true, message: "OK", data: null);
```

25.7 Database helper functions for C# plugins and C# AF extension functions

25.7.1 DBCursor

- **Purpose:** store the query results of CodiacSDKApi.ExecuteSql ; and able to fetch row sequentially from that query results
- **Properties:**
  - Size property refers to number of rows (read only)
  - Position property refers to position of cursor in query results
  - ColumnNames property returns the column names of query results. The column names are in alphabetic order.
- **Methods:**
  - FetchOne method returns row (in list of columns); and next fetch_one() call returns next row. Note that the column names are in alphabetic order.
- **Note:** See the example of CodiacSDKApi.ExecuteSQL function for usage

25.7.2 CodiacSDKApi.ExecuteSQL(sqlQuery, dbConnector)

- **Purpose:** query the database by running SQL query to Codiac database or to other database (with DB connection information)
- **Note:** dbConnector parameter is optional and default to null which means to connect to Codiac database

- **Example:** This example run SQL query to a database (with DB connection information) and dump the first row result.

```
DbConnector dbConnector = new DbConnector()
{
    DbTypeStr = "Postgres",
    DbName = "[redacted]",
    DbUser = "[redacted]",
    DbPass = "[redacted]",
    DbPort = "5432",
    DbAddress = "[redacted]"
};

DbCursor sqlResult = CodiacSDKApi.ExecuteSql("SELECT * FROM \"PolicyAgent\"", dbConnector);

Log($"DB Cursor size: {sqlResult.Size}");

IEnumerable<string> columnNames = sqlResult.ColumnNames;
IEnumerable<string> fetchedSql = sqlResult.FetchOne();

Log("DB fetch one: [" + string.Join(",", fetchedSql) + "]");
Log($"Position after fetching the result: {sqlResult.Position}");

sqlResult.Position = 0;
Log($"Position result: {sqlResult.Position}");
```

25.8 Other CodiacSDK helper functions for C# plugins and C# AF extension functions

25.8.1 CodiacSDKApi.SendEmail(to, subject, body)

- **Purpose:** send an email to recipient with subject and message
- **Example:** This AF extension function example sends an email with recipient from “Alias.EmailSendTests.recipient” alias, subject from “Alias.EmailSendTests.subject” alias and message from “Alias.EmailSendTests.body” alias.

```
AliasValue recipient = GetValueFromMap("Alias.EmailSendTests.recipient");
AliasValue subject = GetValueFromMap("Alias.EmailSendTests.subject");
AliasValue body = GetValueFromMap("Alias.EmailSendTests.body");
CodiacSDKApi.SendEmail(recipient.Value, subject.Value, body.Value);
```

25.8.2 CodiacSDKApi.SendEmail(to, subject, body, , attachments)

- **Purpose:** Sends an email to recipient with subject, message and attachments. The attachment parameter is a delimited string of file container IDs.
- **Return:** A serialized JSON with details.
- **Example:**

```
string to = "slavkopoposki@gmail.com";
string subject = "SUBJECT";
string body = "BODY";

string files = "1178;1179;1180";

// sends an email with attachments, the file that correspond to the id will the attached to the email
CodiacSDKApi.SendEmail(to, subject, body, files);

return DataLoaderHelpers.PrepareResponse(status: true, message: "OK", data: null);
```

25.8.3 CodiacSDKApi.SendSms(to, body)

- **Purpose:** send SMS message to recipient with message
- **Example:** This example send SMS message to 5556789090 (not a real phone number). Only US numbers are accepted.

```
CodiacSDKApi.SendSms("5556789090", "This is a test message.");
```

25.8.4 CodiacSDKApi.GenerateReport(reportName, inlineJson, isSaveToXml)

- **Purpose:** generate report PDF file from report template (client report builder)
- **Note:** The isSaveToXml default to false; if set to true, the report are save to XML file and is not generated to PDF.
- **Example:** This example generate report from report template “Test CrunchF Report” with JSON data from an alias view data.

```
var aliasViewData = jsonParams["alias_view_data"];
string inlineJson = (aliasViewData.Type == JTokenType.Array && aliasViewData.HasValues
    ? aliasViewData.First().ToString(Formatting.None)
    : aliasViewData.ToString(Formatting.None));
return CodiacSDKApi.GenerateReport("Test CrunchF Report", inlineJson);
```

- **Example:** Another example that creates JSON data from aliases.

```
// dependency aliases
AliasValue agent = GetValueFromMap("Alias.Crunch.Agent");
AliasValue owner = GetValueFromMap("Alias.Crunch.Owner");
AliasValue address1 = GetValueFromMap("Alias.Crunch.Address1");
AliasValue address2 = GetValueFromMap("Alias.Crunch.Address2");
AliasValue address3 = GetValueFromMap("Alias.Crunch.Address3");
AliasValue contract = GetValueFromMap("Alias.Crunch.Contract");
AliasValue annuitant = GetValueFromMap("Alias.Crunch.Annuitant");
AliasValue maturityDate = GetValueFromMap("Alias.Crunch.MaturityDate");

// list of aliases in json (in Alias View JSON format)
JObject jsonInput = new JObject
{
    { "Alias.Crunch", new JObject
        {
            { "Alias.Crunch.Agent", agent?.Value },
            { "Alias.Crunch.Owner", owner?.Value },
            { "Alias.Crunch.Address1", address1?.Value },
            { "Alias.Crunch.Address2", address2?.Value },
            { "Alias.Crunch.Address3", address3?.Value },
            { "Alias.Crunch.Contract", contract?.Value },
            { "Alias.Crunch.Annuitant", annuitant?.Value },
            { "Alias.Crunch.MaturityDate", maturityDate?.Value },
        }
    }
};

// generate report
CodiacSDKApi.GenerateReport("Test Crunch Report", jsonInput.ToString());
```

25.8.5 CodiacSDKApi.GenerateReport(reportName, inlineData, delimiter, isSaveToXml)

- **Purpose:** generate report PDF file from report template (client report builder)
- **Note:** The isSaveToXml default to false; if set to true, the report are save to XML file and is not generated to PDF.
- **Example:** This example generate report from report template “Test CrunchF Report” with values from vector of delimited values. The delimiter used '|'.

```
string[] inlineData =
{
    "Contract|FundNo|Description|Amount",
    "Contract A|CA1|Test contract A - 1|123.45",
    "Contract A|CA2|Test contract A - 2|3.45",
    "Contract B|CB1|Test contract B - 1|645",
    "Contract C|CC1|Test contract C - 1|7.89"
};

// generate report
CodiacSDKApi.GenerateReport("Test CrunchF Report - inline data", inlineData, delimiter: "|");
```

25.8.6 CodiacSDKApi.ExecuteJson(jsonRequest)

- **Purpose:** call Codiac service by executing JSON request
- **Example:** This example calls GetJobsScheduleList function of JobsScheduler service to get list of jobs. A JSON request is created with func_name parameter and lib_name parameter.

```
// create json request
JObject request = new JObject
{
    { "requestbody", new JObject
        {
            { "func_name", "GetJobScheduleList" },
            { "lib_name", "CodiacSDK.zJobsScheduler" }
        }
    }
};

// execute service to get Jobs
JObject response = CodiacSDKApi.ExecuteJson(request);
```

- **Note:** The user\session parameters are filled up at service process.

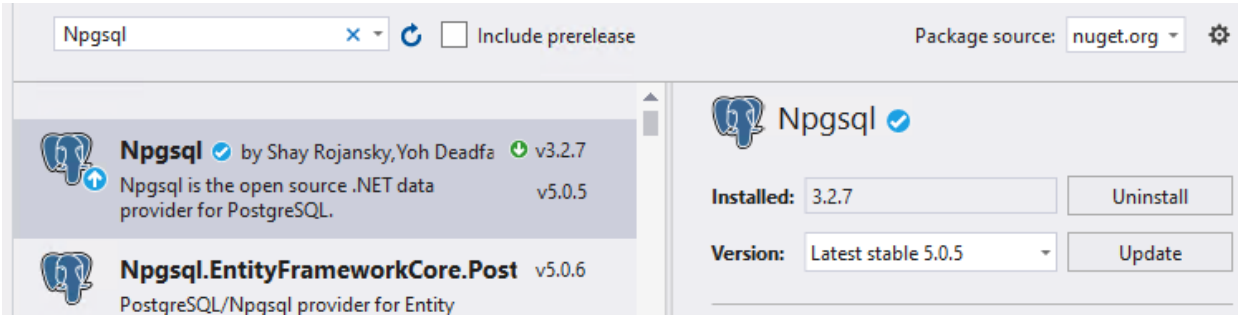
25.9 SQL query with C# native database modules

Codiac SDK provides database helper functions that can access Codiac database and other databases. In addition, the user can add database module to plugin\module project with NuGet package manager.

The DirectSql project (C# plugin example project) contains example source code on how to query the databases with native database modules.

- **PostgreSQL**

  The npgsql module can be use to query PostgreSQL database. To add this module to your project, go to NuGet Package Manager and install Npgsql 3.2.7 which is for Net Framework 4.5.2



  Below is the sequence for npgsql module to query database:

  1. Create “NpgsqlConnection” object with database connection string
  2. Call “OpenAsync” function asynchronously on “NpgsqlConnection” object

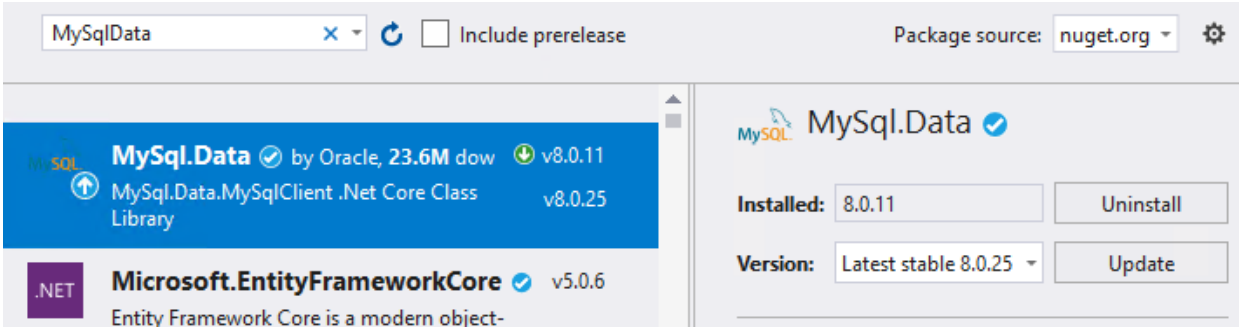
3. Create “NpgsqlCommand” object with SQL query command
4. For SQL query without result,
   a) Call “ExecuteNonQueryAsync” function on “NpgsqlCommand” object
5. For SQL query with results,
   a) Call “ExecuteReaderAsync” function asynchronously on “NpgsqlCommand” object
      - This returns a reader object.
   b) Call “ReadAsync” function on reader object to fetch query result

See the PostgreSqlAsync function of DirectSql project for full example.

See the https://www.npgsql.org/doc/index.html for full details on how to use npgsql module.

- **MySQL**

  The MySql.Data module by Oracle can be use to query MySQL database. To add this module to your project, go to NuGet Package Manager and install MySQL.data v8.0.11 which is for Net Framework 4.5.2.



Below is the sequence for MySql.Data module to query database:

1. Create “MySqlConnection” object with database connection string
2. Call “Open” function on “MySqlConnection” object
3. Create “MySqlCommand” object with SQL query command
4. For SQL query without result,
   a) Call “ExecuteNonQuery” function on “MySqlCommand” object
5. For SQL query with results,
   a) Call “ExecuteReader” function on “MySqlCommand” object
      - This returns a reader object.
   b) Call “Read” function on reader object to fetch query result

See the MySqlSync function of DirectSql project for full example.

See the https://dev.mysql.com/doc/connector-net/en/connector-net-programming.html for

full details on how to use MySql.Data module.

- **MS SQL Server**

  The built-in System.Data.SqlClient of Net Framework can be use to query MS SQL Server database.

```
using System.Data.SqlClient;
```

  Below is the sequence for MySql.Data module to query database:

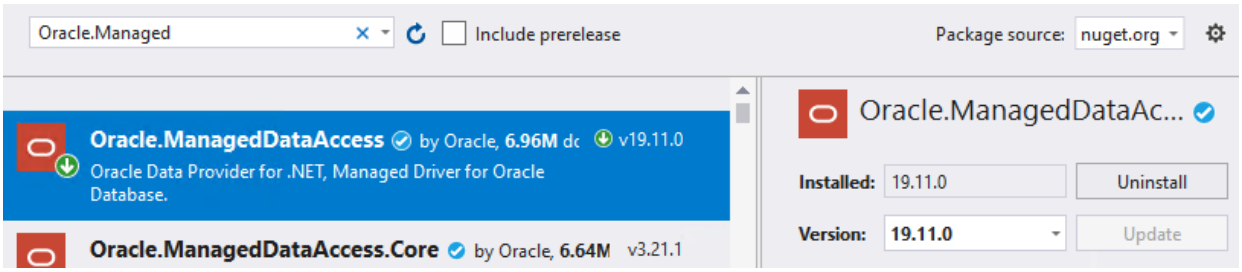
  1. Create “SqlConnection” object with database connection string
  2. Call “Open” function on “SqlConnection” object
  3. Create “SqlCommand” object with SQL query command
  4. For SQL query without result,
     a) Call “ExecuteNonQuery” function on “SqlCommand” object
  5. For SQL query with results,
     a) Call “ExecuteReader” function on “SqlCommand” object
        - This returns a reader object.
     b) Call “Read” function on reader object to fetch query result

  See the MsSqlServerSync function of DirectSql project for full example.

  See the Microsoft Documentation (https://docs.microsoft.com/en-us/documentation) to find out on how to use SqlClient APIs.

- **Oracle**

  The Oracle.ManagedDataAccess module can be use to query Oracle database. To add this module to your project, go to NuGet Package Manager and install Oracle.ManagedDataAccess v19.11.0.



  Below is the sequence for Oracle.ManagedDataAccess module to query database:

  1. Create “OracleConnection” object with database connection string
  2. Call “Open” function on “OracleConnection” object
  3. Create “OracleCommand” object with SQL query command
  4. For SQL query without result,

a) Call “ExecuteNonQuery” function on “OracleCommand” object

5. For SQL query with results,
   a) Call “ExecuteReader” function on “OracleCommand” object
      - This returns a reader object.
   b) Call “Read” function on reader object to fetch query result

See the OracleSqlSync function of DirectSql project for full example.

See the Oracle Documentation (https://docs.oracle.com/en/database/oracle/oracle-data-access-components/index.html) to find out on how to use Oracle.ManagedDataAccess APIs.

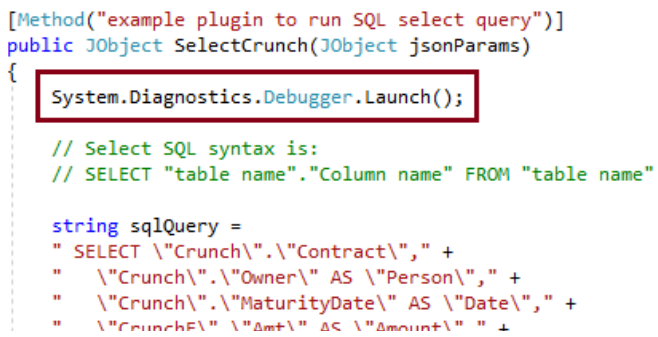
26 Debugging C# plugin or AF extension module

Prerequisite before staring the debugger:

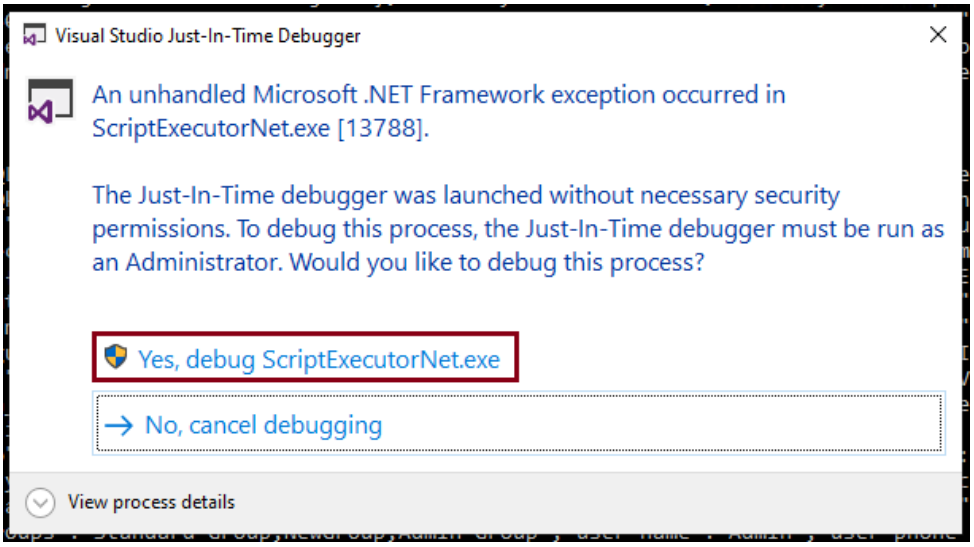
- the plugin or module are build with debug symbols
- the plugin (DLL or so) file to be debugged is in *<service directory>*\plugins\net directory
- the module (DLL or so) file to be debugged is in *<service directory>*\Modules\net directory
- service is running

Steps to run Visual Studio debugger:

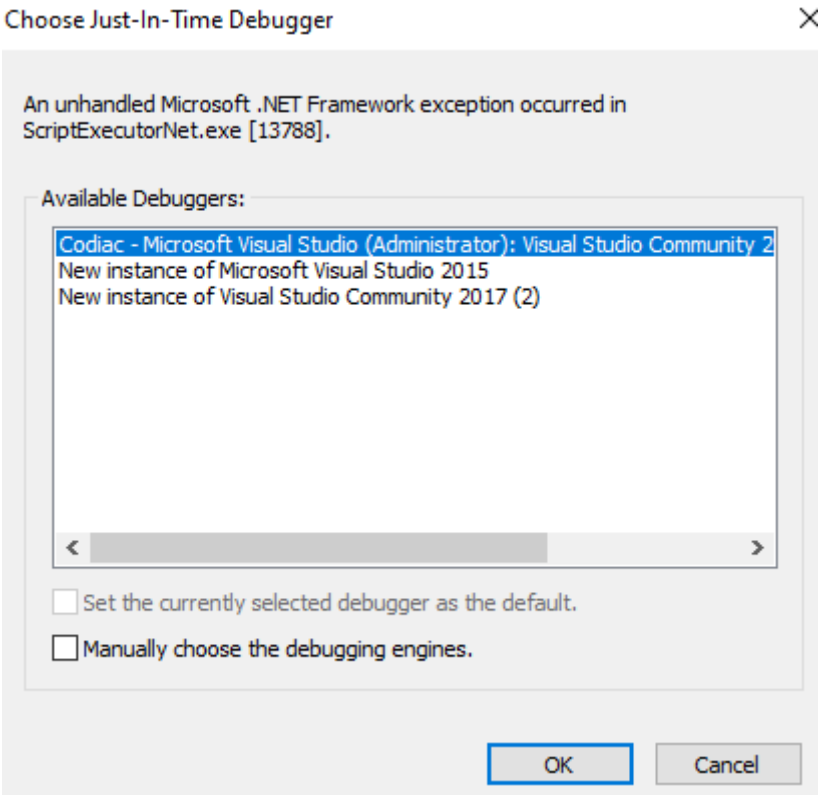
1. Insert `System.Diagnostics.Debugger.Launch();` to where you want to start the debug. This will invoke debugger at run-time.



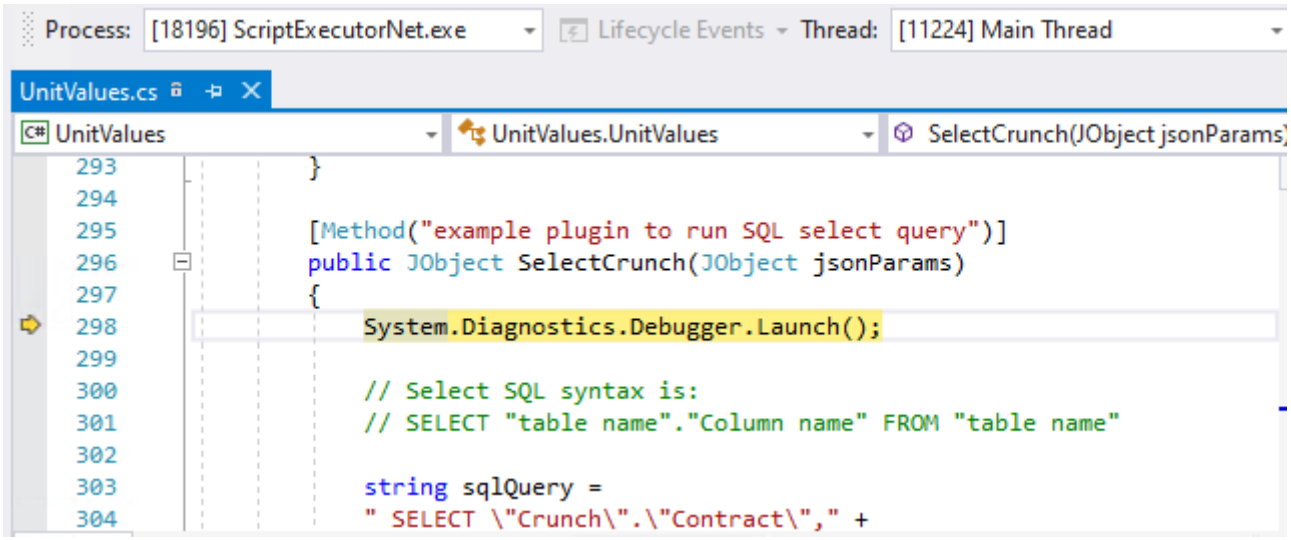
2. Build plugin or AF extension module and copy the output DLL files to *<service directory>*\plugins\net directory (if plugin) or *<service directory>*\Modules\net directory (if AF extension module)
3. Execute the plugin or module function. This will prompt a Just-In-Time debugger window. Click **Yes**.



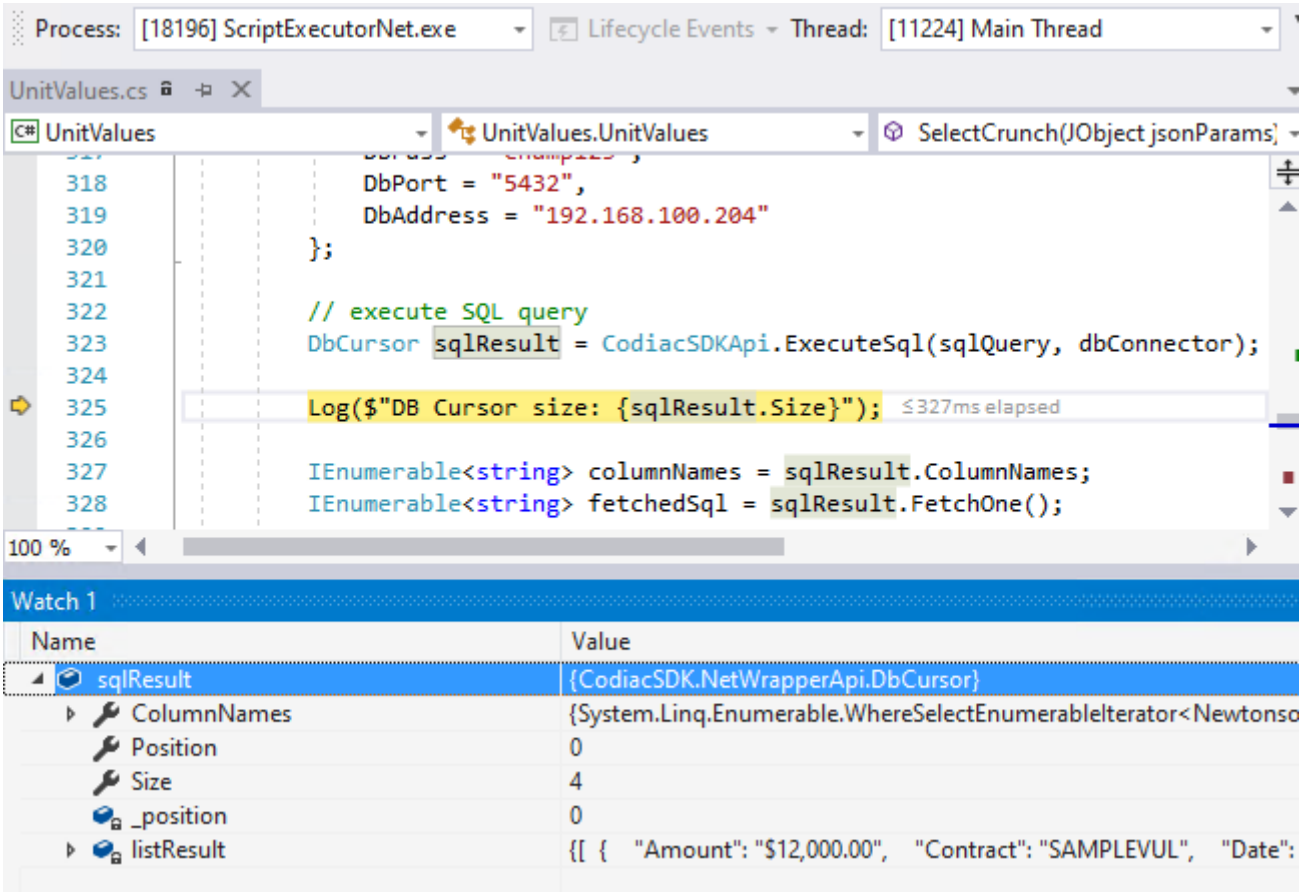
4. Select a Visual Studio debugger instance.



5. At this point, the debugger should stop at `Debugger.Launch()`.



6. You can step thru the code (**F10** – step over, **F11** – step into) and see the variable values in watches.



7. To end debugging, go to **Debug** menu and select **Detach all**.

27 Remote Debugging C# plugin and AF extension functions

Full instruction for the remote debugging for the C# applications can be found on the Remote Debug a C++ Project - Visual Studio (Windows) | Microsoft Learn page.

27.1 Required tools

Download and install the tools below on the remote machine:

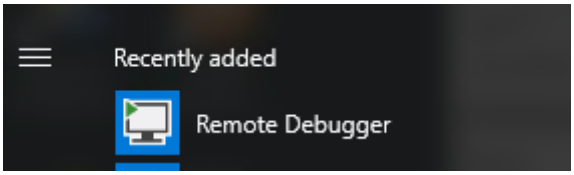
- **Remote Debugger**
  The Remote Debugger can be downloaded at the following link: Remote Debug a C++ Project - Visual Studio (Windows) | Microsoft Learn

27.2 Tool configurations

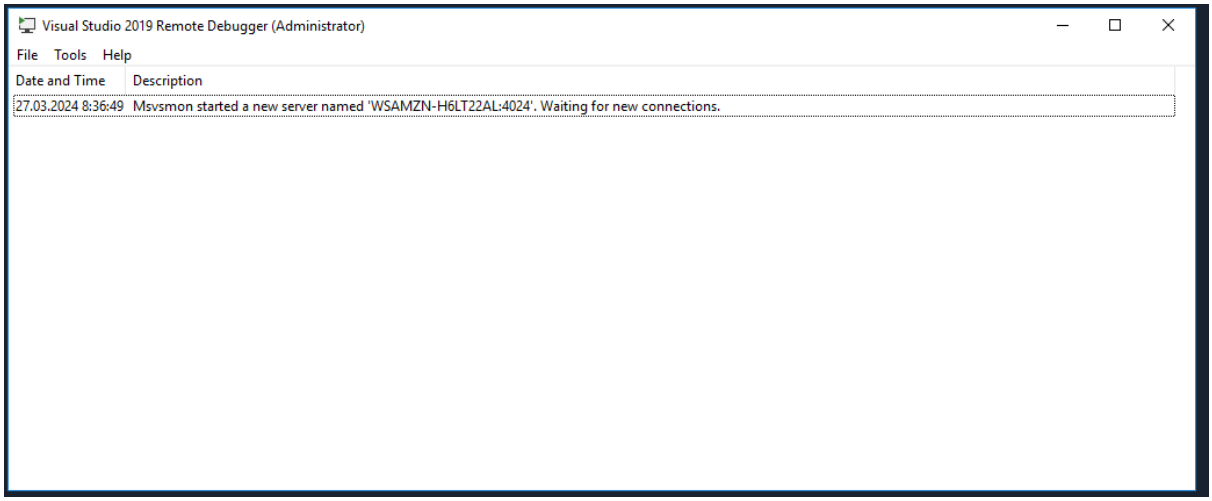
27.2.1 Install Remote Debugger

To set up the tool for remote debugging, follow the steps below:

1. Download the Remote Debugger.
2. Install the Remote Debugger on the remote machine.
3. Launch the Remote Debugger on the remote machine.

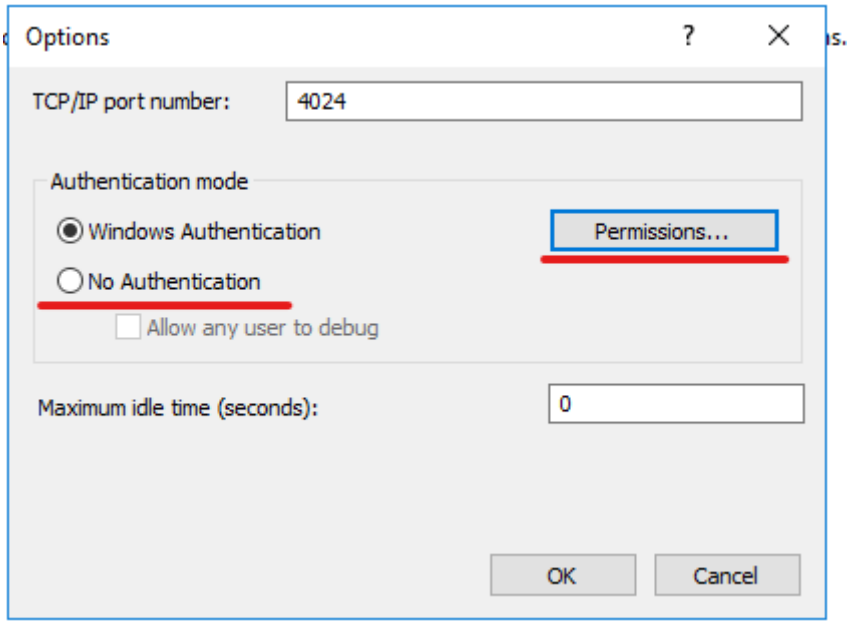


The Remote Debugger working screen will be displayed as follows:



In the Remote Debugger working screen, perform the following actions, to configure the access to the Remote Debugger:

1. In the Menu, click the **Tools** menu section and select the **Options…** menu item. The Options pop-up window opens.



In the Options pop-up window, fill in the necessary fields:

- **TCP/IP port number** – the communication endpoints that applications use to coordinate and establish communication over a network using the TCP/IP protocol suite.

Based on the desired security level select:

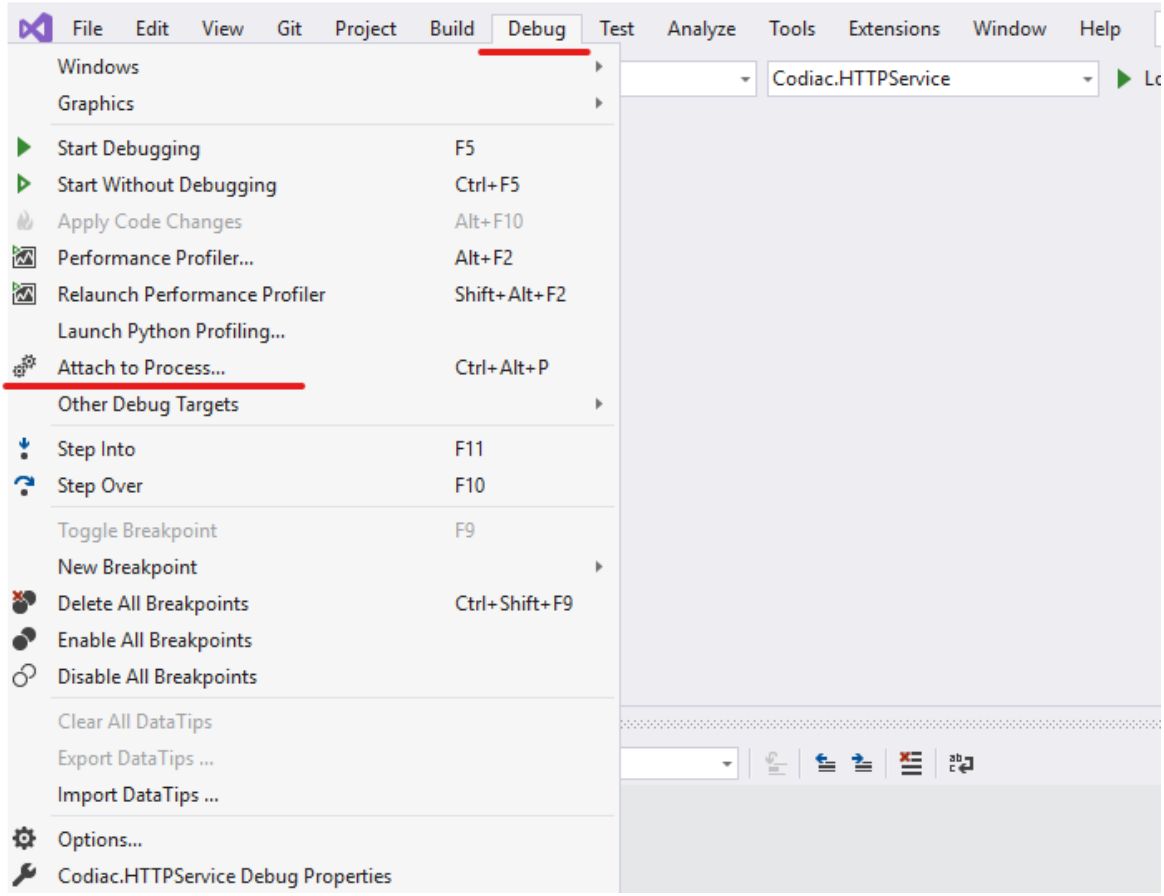
- the **Windows Authentication** access and set the **Permissions** to a user,
  or
- the **No Authentication** access for the remote debugging.

After the above-mentioned options are set, the remote machine will be ready to accept the Remote Debugger's requests.

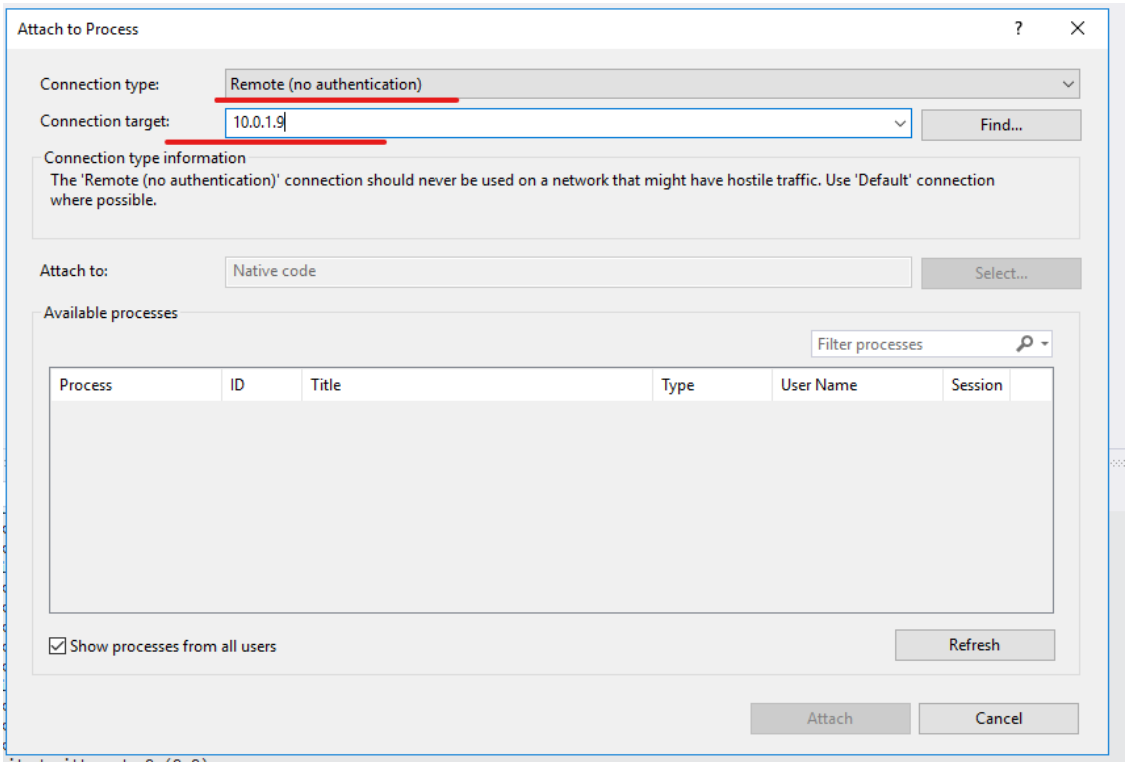
27.3 Debugging

On the local machine where plugins or modules were created, users just need to select the process in the Visual Studio. The Remote Debugger will be attached to the selected process. For that:

1. Launch Visual Studio.
2. Click the **Debug** section on the menu, and select the **Attach to Process…** menu item.



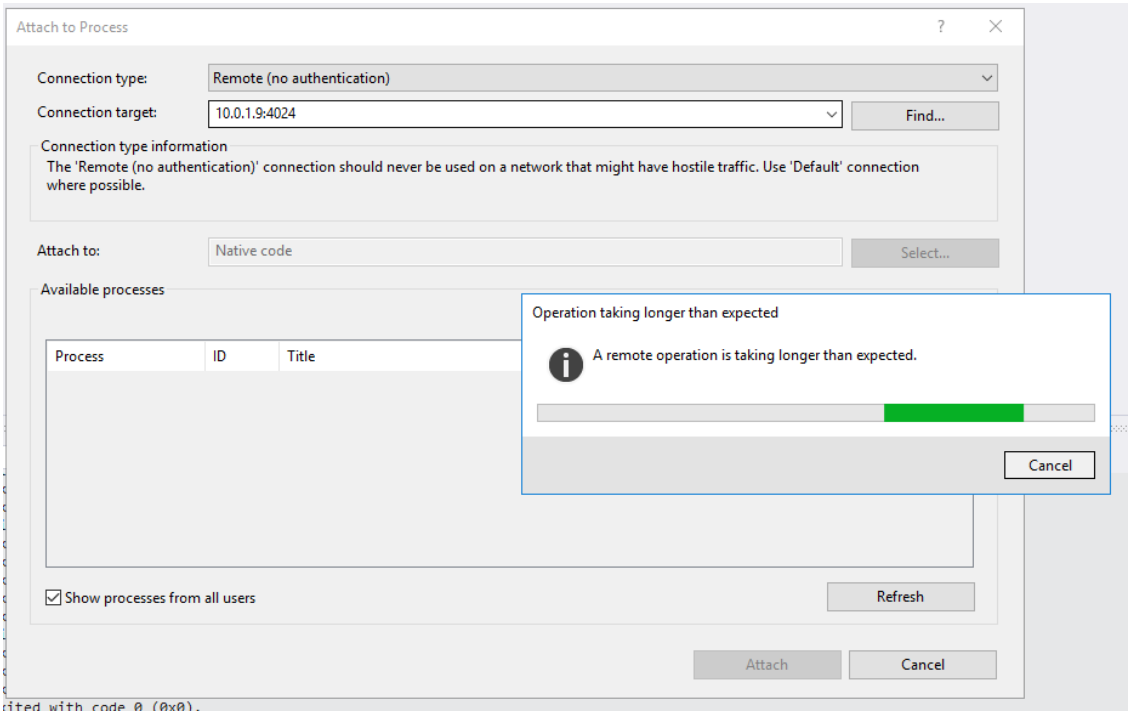
The **Attach to Process** pop-up window opens.



3. In the opened **Attach to Process** pop-up window, select:
   - the **Remote (No authentication)** debugger type from the Connection type drop-down list,
   - the remote machine IP address from the Connection target drop-down list, and

- press the ENTER key.

The debugger will connect to the remote machine, and the list of available processes that the debugger can be attached to will be shown.



4. In the available processes list, select the **Codiac.HTTPService.exe** process, and click the **Attach** button.