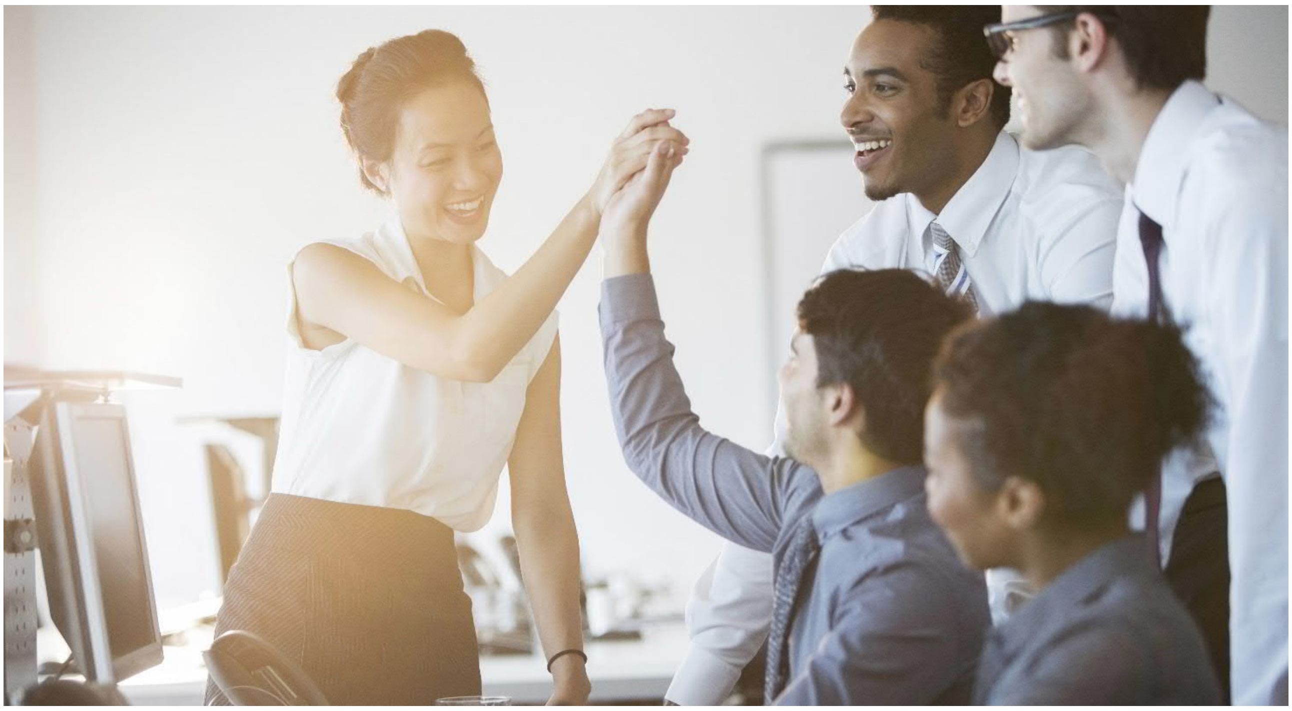


Exercise Solution: Establish a Connection to SAP Business One

User Interface API

PUBLIC

INTRODUCTION

In this exercise, you will perform the following tasks:

1. Create a new Visual Studio project
2. Implement a connection to a running SAP Business One application
3. Display a *MessageBox* within SAP Business One
4. Connect to the SAP Business One by using the Single-Sign-On feature
5. Connect to the SAP Business One by using Multiple Add-On feature
6. Define the AppEvent handler

PREREQUISITE:

- This document is using the **C Sharp** (C#) language
- This document is using the Microsoft Visual Studio 2015
- Use the demo database for SAP Business One, version for SAP HANA or SAP Business One
- Credentials: User code: **manager**

GUIDELINES:

The screenshots provided here are for your reference only and may differ from the actual screenshots in your system.

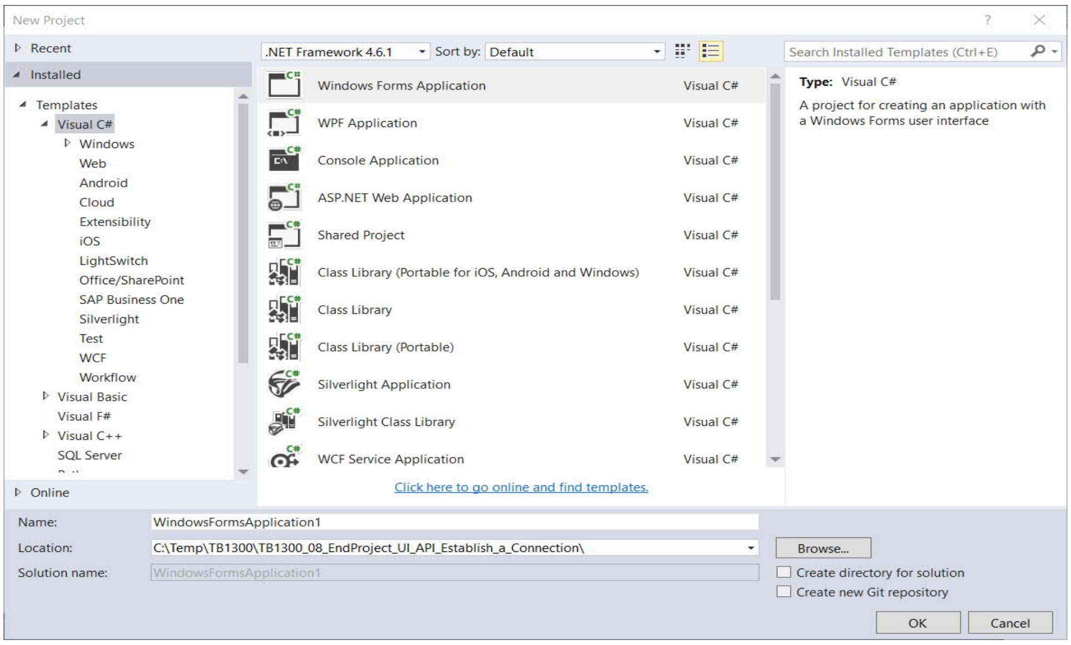
1. TASK - CREATE A NEW VISUAL STUDIO PROJECT

1.1. Install the SAP Business One Software Development Kit, available on the product CD

1.2. Create a new Visual Studio project.

1.2.1. Open Visual Studio and choose *File → New → Project*

1.2.2. Select the **Windows Forms Application** template from *Installed → Templates → Windows* section.



New Project

Type: Visual C#

A project for creating an application with a Windows Forms user interface

Name: WindowsFormsApplication1

Location: C:\Temp\TB1300\TB1300_08_EndProject_UI_API_Establish_a_Connection\

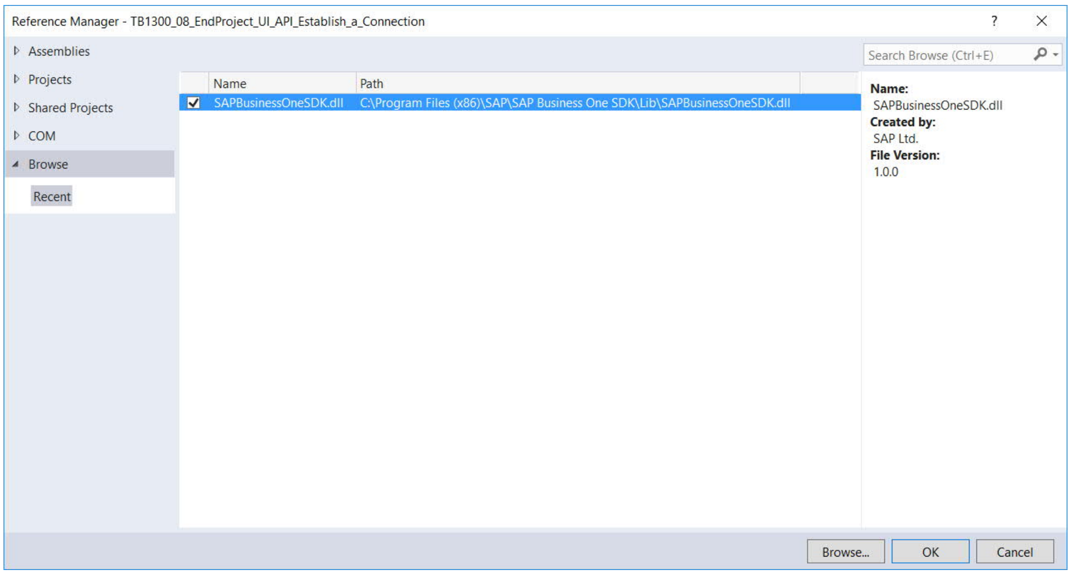
Solution name: WindowsFormsApplication1

Recent
Installed
Templates
Visual C#
Windows
Web
Android
Cloud
Extensibility
iOS
LightSwitch
Office/SharePoint
SAP Business One
Silverlight
Test
WCF
Workflow
Visual Basic
Visual F#
Visual C++
SQL Server
Online
.NET Framework 4.6.1
Sort by: Default
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
ASP.NET Web Application Visual C#
Shared Project Visual C#
Class Library (Portable for iOS, Android and Windows) Visual C#
Class Library Visual C#
Class Library (Portable) Visual C#
Silverlight Application Visual C#
Silverlight Class Library Visual C#
WCF Service Application Visual C#
Click here to go online and find templates.
Search Installed Templates (Ctrl+E)
Browse...
Create directory for solution
Create new Git repository
OK
Cancel

1.3. Add the reference to the SAPBusinessOneSDK.dll

1.3.1. Select the main menu *Project → Add Reference*

1.3.2. Click to the *Browse* button at the bottom of the form → Browse for file **SAPBusinessOneSDK.dll** (default location is *c:\Program Files (x86)\SAP\SAP Business One SDK\Lib\*)



Reference Manager - TB1300_08_EndProject_UI_API_Establish_a_Connection

Name	Path
✓ SAPBusinessOneSDK.dll	C:\Program Files (x86)\SAP\SAP Business One SDK\Lib\SAPBusinessOneSDK.dll

Name: SAPBusinessOneSDK.dll
Created by: SAP Ltd.
File Version: 1.0.0

1.4. Remove the Form from the solution and fix the dependencies.

```
static void Main()
        {
            Application.Run();

        }
```

2. TASK - IMPLEMENT A CONNECTION TO A RUNNING SAP BUSINESS ONE APPLICATION

2.1. Define the variables you need for a connection to a running SAP Business One application.

2.1.1. Define the global variable for *Application* object.

```
private static SAPbouiCOM.Application SBO_Application;
```

2.1.2. Create a new function for connecting to the UI API, call it **ConnectToUI**. In the function define the SboGuiApi variable and the Connection string

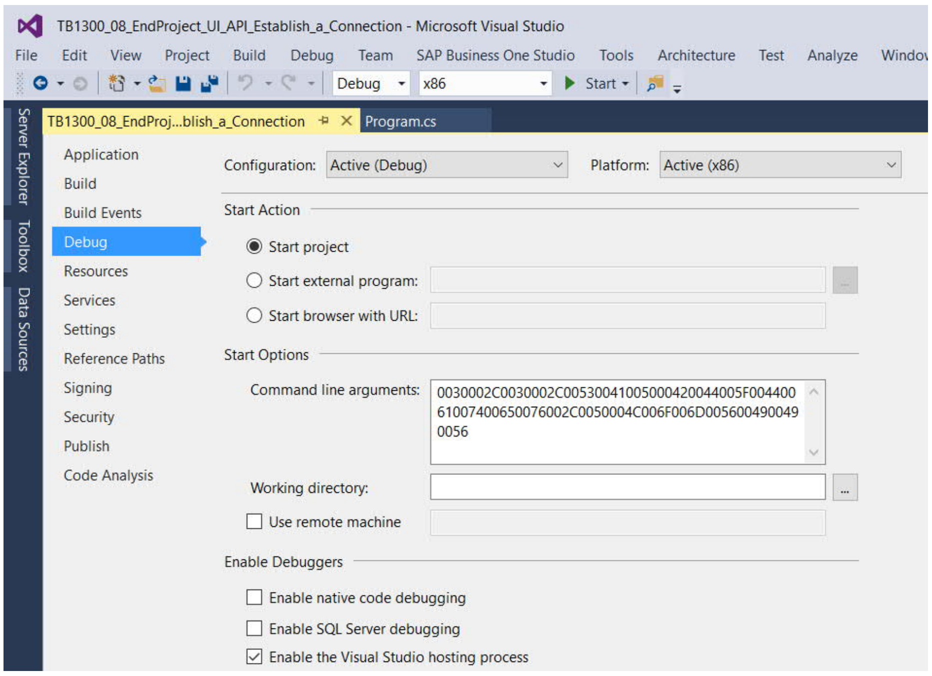
```
SAPbouiCOM.SboGuiApi SboGuiApi;

 string sConnectionString;
```

2.2. Connect to the SAP Business One SboGuiApi and get a handle to the running application.

2.2.1.Enter the connection string value as a Command Line argument

Project → Properties → Debug → Command Line arguments


TB1300_08_EndProject_UI_API_Establish_a_Connection - Microsoft Visual Studio
File Edit View Project Build Debug Team SAP Business One Studio Tools Architecture Test Analyze Window
Debug x86 Start
Server Explorer Toolbox Data Sources
TB1300_08_EndProj...blish_a_Connection Program.cs
Application
Build
Build Events
Debug
Resources
Services
Settings
Reference Paths
Signing
Security
Publish
Code Analysis

Configuration: Active (Debug)   Platform: Active (x86)

Start Action

[✓ Start project]
Start external program:
Start browser with URL:

Start Options

Command line arguments: 0030002C0030002C00530041005000420044005F004400
61007400650076002C0050004C006F006D005600490049
0056

Working directory:

☐ Use remote machine

Enable Debuggers

☐ Enable native code debugging
☐ Enable SQL Server debugging
☑ Enable the Visual Studio hosting process

sConnectionString = "0030002C0030002C00530041005000420044005F0044006100740065007600 2C0050004C006F006D0056004900490056"

2.2.2. Connect to the SAP Business One *SboGuiApi* in function **ConnectToUI**

```
SboGuiApi = new SAPbouiCOM.SboGuiApi();
sConnectionString =
System.Convert.ToString(Environment.GetCommandLineArgs().GetValue(1));

SboGuiApi.Connect(sConnectionString);
SBO_Application = SboGuiApi.GetApplication();
```

3. TASK - DISPLAY A MESSAGEBOX WITHIN SAP BUSINESS ONE

3.1. The method to display a MessageBox has several optional parameters. Check them out.

There is a Method of the Application object to display message boxes within SAP Business One

```
SBO_Application.MessageBox("Connected to UI API", 1, "Continue", "Cancel");
```

4. TASK - CONNECT TO THE SAP BUSINESS ONE BY USING THE SINGLE-SIGN-ON FEATURE

4.1. Define the global variable for *Application* object.

```
private static SAPbobsCOM.Company diCompany;
```

4.2. Create a new function, call it **ConnectwithSSO**, which will perform the DI API connection by using the cookie information get from the DI API's *GetContextCookie* method

```
private static void ConnectwithSSO()
{
    diCompany = new SAPbobsCOM.Company();
    string cookie = diCompany.GetContextCookie();
    string connInfo = SBO_Application.Company.GetConnectionContext(cookie);

    int ret = diCompany.SetSboLoginContext(connInfo);
    if (ret != 0)
        SBO_Application.MessageBox("DI Connection failed!", 0, "Ok", "", "");
    else
        SBO_Application.MessageBox("Connected with SSO!", 0, "Ok", "", "");
}
```

4.3. Call the function **ConnectwithSSO** in the function **ConnectToUI**

```
ConnectwithSSO();
```

5. TASK - CONNECT TO THE SAP BUSINESS ONE BY USING MULTIPLE ADD-ON FEATURE

5.1. Comment the function call **ConnectwithSSO** in the function **ConnectToUI**

```
//ConnectwithSSO();
```

5.2. Create a new function, call it **ConnectwithSharedMemory**, which will perform the DI API connection by using the *GetDICompany* method. This method will reduce memory consumption when running multiple DI API add-ons.

```
private static void ConnectwithSharedMemory()
        {
            diCompany =
(SAPbobsCOM.Company)Program.SBO_Application.Company.GetDICompany();
            SBO_Application.MessageBox("DI Connected To: " +
Program.diCompany.CompanyName, 0, "Ok", "", "");

        }
```

5.3. Call the function **ConnectwithSharedMemory** in the function **ConnectToUI**

```
ConnectwithSharedMemory();
```

6. TASK – DEFINE THE APPEVENT HANDLER

6.1. Create a function which will handle the mandatory AppEvent.

```
public static void SBO_Application_AppEvent(SAPbouiCOM.BoAppEventTypes EventType)
        {
            switch (EventType)
            {
                case SAPbouiCOM.BoAppEventTypes.aet_ShutDown:
                    //Exit Add-On
                    SBO_Application.MessageBox("My is addon disconnected." +
Program.diCompany.CompanyName, 0, "Ok", "", "");
                    System.Runtime.InteropServices.Marshal.ReleaseComObject(Program.d
                    iCompany);
                    Application.Exit();
                    break;
                case SAPbouiCOM.BoAppEventTypes.aet_CompanyChanged:
                    break;
                case SAPbouiCOM.BoAppEventTypes.aet_FontChanged:
                    break;
                case SAPbouiCOM.BoAppEventTypes.aet_LanguageChanged:
                    break;
                case SAPbouiCOM.BoAppEventTypes.aet_ServerTerminition:
                    break;
                default:
                    break;
            }

        }
```

6.2. Create event subscriber for the previously create function before the application run will be called in the **Main** function.

```
SBO_Application.AppEvent += new
SAPbouiCOM._IApplicationEvents_AppEventEventHandler(SBO_Application_AppEvent);
```

Solutions can be found in the SDK Help Center documentation and SDK samples (in the SDK Folder – see Appendix “SDK Installations” for more information),
COM UI / CSharp / 01.HelloWorld
COM UI / CSharp / 02.CatchingEvents
COM UI DI / CSharp / Hello World

www.sap.com