Technical Note

Using ClientAce® in Visual Studio

1. Introduction

This document aims to instruct knowledgeable developers on the steps that can be performed to use the latest version of ClientAce 4 in the Visual Studio 2015, 2017, or 2019 development environment.

The current install of ClientAce does not detect these versions of Visual Studio but the ClientAce API, controls, and components are compatible. Users can manually include the API and components in a project and manually add the controls to the Toolbox. There will be no sign buttons to sign a project to run in release mode; however, the post-build event commands for signing can be added manually.

The following techniques can be applied with ClientAce in Visual Studio 2015, 2017 or 2019.

2. Getting ClientAce 4 on a PC with Visual Studio Installed

The current ClientAce install does not detect the Integrated Development Environment (IDE) for Visual Studio 2015, 2017 or 2019. There are a couple of ways to install ClientAce on a PC.

2.1 ClientAce Installed in Visual Studio 2010 to 2013

If a supported version of Visual Studio is installed with ClientAce, Visual Studio 2015 can be installed as well. Multiple versions of Visual Studio can be installed at the same time.

If the only version currently installed is Visual Studio 2015 or later, install Visual Studio 2013 Community Edition first. Then ClientAce install detects it and completes the installation successfully.

2.2 Copy ClientAce from a PC with a Current Install

Some situations or business rules make it impossible to install multiple versions of Visual Studio on a single PC. In those cases, either copy the ClientAce install folder from a PC where it is already installed or, if it has never been installed, get a copy of the installation folder from Kepware Technical Support.

On 32-bit operating systems, ClientAce installs, by default, to:
`C:\Program Files\Kepware Technologies\ClientAce`

On 64-bit operating systems, ClientAce installs, by default, to:
`C:\Program Files (x86)\Kepware Technologies\ClientAce`

- **Note**: Copy the entire ClientAce folder from its installed location to the new PC.

3. Using ClientAce in Visual Studio 2015, 2017, or 2019

Once the ClientAce files are on a PC with Visual Studio 2015 or later, a few components need to be configured.

3.1 License the Sign Tool

These instructions assume the user has a valid ClientAce license.

If ClientAce is installed with another version of Visual Studio, use the Command prompt to license the Sign Tool.

If you moved the Client Ace folder from another PC, it must be licensed on the new PC.

1. Launch the Command prompt.

- **Note**: Run the Command prompt as an Administrator.

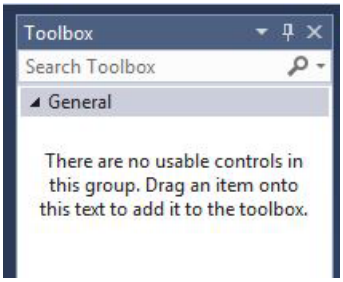
2. In the Command window, type the following:

```
"C:\Program Files (x86)\Kepware Technologies\ClientAce\Sign\sign_gui.exe" -register
```

3. This opens the dialog to generate the License Request, which must be sent to Kepware. Follow the prompts to complete the licensing process.
4. Once complete, created applications can be signed.

3.2 Add Controls to the Toolbox

Since the install does not detect Visual Studio, there are no Toolbox items for the controls. They must be added manually through the following steps.



1. Open Visual Studio.
2. Create a VB.Net or C# project.
3. Expand the toolbox and pin it open.
4. Right-click the Toolbox and select **Add Tab...**.

5. Name the new tab something that identifies the tab as controls for ClientAce Windows Forms. For example: "ClientAce 4.0(WF)"
6. With the new tab selected, right-click and select **Choose Items**.
7. In the Choose Toolbox Items dialog, click **Browse...**.
8. In the Browse dialog, locate the ClientAce\V4.0 folder.
9. Select the following controls:
    - Kepware.ClientAce.BrowseControls.dll
    - Kepware.ClientAce.DAJunction.dll
    - Kepware.ClientAce.KEPServerEXControlls.dll
10. Click **Open**.
11. In the Choose Toolbox Items dialog, click **OK** to add the new items to the Toolbox.

3.3 Reference the API in the Project

When dragging the controls to the forms in a project, they are added to its references. However, if writing a full application and not using any of the controls; the reference for the API must be added manually through the following steps.

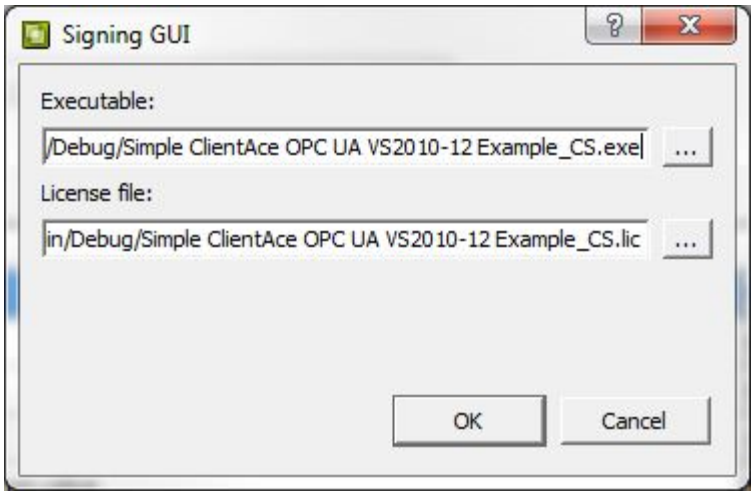
1. Right-click **References** in the Solution Explorer and select **Add Reference**.
2. In the Reference Manager dialog, click **Browse** and locate the ClientAce\v4.0 folder.
3. Select **Kepware.ClientAce.OPCClient.dll**.
4. Click **Add**.
5. In the Reference Manager dialog, click **OK** to add the reference to the project.

3.4 Sign the Project

Signing the project allows it to run without launching into demo mode. In typical installs, click the **Sign** button in the Visual Studio button menu to add post-build event commands to sign the application. Since the **Sign** button is not available in Visual Studio 2015 at this time, there are two options explained in the ClientAce help file briefly discussed below.

3.4.1 Manually Sign the Application After Compile

1. Access the Start menu and navigate to **Programs | Kepware Products**.
2. Select **ClientAce | Sign Executable** to open the **Signing GUI** dialog.
3. Click the **Ellipses (...)** button to browse to the folder with the compiled ClientAce application executable.
4. Select the executable to be signed.
5. Once selected, the named license file appears in the License File field.



6. Click **OK** to complete the generation of the file.

3.4.2 Manually Add the Post-Build Commands

1. In VB.Net, open the Project Properties.
2. Select the **Compile** tab and click on **Build Events**.
3. In C#, open the Project Properties.
4. Select the **Build Events** tab.
5. In both C# and VB.Net, add the command lines below to the **Post-build event command line** section. Once entered, the project will run the command lines after each success full build.

```
"C:\Program Files\Kepware Technologies\ClientAce\Sign\sign.exe" "$(TargetPath)"
"$(TargetName).lic"


"C:\Program Files\Kepware Technologies\ClientAce\Sign\sign.exe" "$(TargetDir)
$(TargetName).vshost.exe" "$(TargetName).vshost.lic"
```

Cautions:

- In Visual Studio 2017 and 2019, Microsoft is no longer supporting VSHost Debugging. When compiling / building an application in that Visual Studio version, only the first command line is added to the post-build event.
- If the target project is being compiled to run as a service, VSHost Debugging must be disabled because it is not supported for service applications. Only the first command line is used in this situation as well.

