

---

ComponentOne

# **ComboBox for ASP.NET**

## **Wijmo**

Copyright © 2012 ComponentOne LLC. All rights reserved.

*Corporate Headquarters*

**ComponentOne LLC**

201 South Highland Avenue  
3<sup>rd</sup> Floor  
Pittsburgh, PA 15206 • USA

**Internet:** [info@ComponentOne.com](mailto:info@ComponentOne.com)

**Web site:** <http://www.componentone.com>

**Sales**

E-mail: [sales@componentone.com](mailto:sales@componentone.com)

Telephone: 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

**Trademarks**

The ComponentOne product name is a trademark and ComponentOne is a registered trademark of ComponentOne LLC. All other trademarks used herein are the properties of their respective owners.

**Warranty**

ComponentOne warrants that the original CD (or diskettes) are free from defects in material and workmanship, assuming normal use, for a period of 90 days from the date of purchase. If a defect occurs during this time, you may return the defective CD (or disk) to ComponentOne, along with a dated proof of purchase, and ComponentOne will replace it at no charge. After 90 days, you can obtain a replacement for a defective CD (or disk) by sending it and a check for \$25 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original CD (or disks) set forth here, ComponentOne makes no other warranties, express or implied. Every attempt has been made to ensure that the information contained in this manual is correct as of the time it was written. We are not responsible for any errors or omissions. ComponentOne's liability is limited to the amount you paid for the product. ComponentOne is not liable for any special, consequential, or other damages for any reason.

**Copying and Distribution**

While you are welcome to make backup copies of the software for your own use and protection, you are not permitted to make copies for the use of anyone else. We put a lot of time and effort into creating this product, and we appreciate your support in seeing that it is used by licensed users only.

This manual was produced using [ComponentOne Doc-To-Help™](#).

# Table of Contents

ComponentOne ComboBox for ASP.NET Wijmo Overview .....	1
Installing Studio for ASP.NET Wijmo .....	1
Studio for ASP.NET Wijmo Setup Files .....	1
System Requirements .....	2
Uninstalling Studio for ASP.NET Wijmo .....	2
Deploying your Application in a Medium Trust Environment .....	2
End-User License Agreement .....	6
Licensing FAQs .....	6
What is Licensing? .....	6
How does Licensing Work? .....	6
Common Scenarios .....	7
Troubleshooting .....	9
Technical Support .....	10
Redistributable Files .....	11
About This Documentation .....	11
Namespaces .....	12
Creating an ASP.NET Project .....	13
Adding the ComboBox for ASP.NET Wijmo Component to a Project .....	14
Key Features .....	17
ComboBox for ASP.NET Wijmo Quick Start .....	17
Step 1 of 4: Creating an Application .....	17
Step 2 of 4: Adding Items to C1ComboBox .....	18
Step 3 of 4: Create an Event Handler for the Selected Item .....	18
Step 4 of 4: Running the Project .....	19
Wijmo Top Tips .....	19
Design-Time Support .....	20
C1ComboBox Smart Tag .....	20
C1ComboBox Context Menu .....	21
C1ComboBox Collection Editors .....	21
C1ComboBox Editor .....	21

Columns Designer .....	22
ComboBox Appearance .....	23
C1ComboBox CSS Selectors .....	24
Themes .....	25
Working with the Client-Side ComboBox.....	26
Client-Side Events.....	26
ComboBox for ASP.NET Wijmo Samples .....	27
ComboBox for ASP.NET Wijmo Task-Based Help .....	29
Adding a Custom Theme .....	29
Combobox for ASP.NET Wijmo Client-Side Reference .....	31
Using the Wijmo CDN .....	31

# ComponentOne ComboBox for ASP.NET Wijmo Overview

**ComponentOne ComboBox™ for ASP.NET Wijmo** is a full-featured combo box control that combines an editable text box with an auto-searchable drop-down list.

For a list of the latest features added to **ComponentOne Studio for ASP.NET Wijmo**, visit [What's New in Studio for ASP.NET Wijmo](#).

## Getting Started

To get started, review the following topics:

- [Key Features](#) (page 17)
- [ComboBox for ASP.NET Wijmo Quick Start](#) (page 17)
- [ComboBox for ASP.NET Wijmo Samples](#) (page 27)

## Installing Studio for ASP.NET Wijmo

The following sections provide helpful information on installing **ComponentOne Studio for ASP.NET Wijmo**:

### Studio for ASP.NET Wijmo Setup Files

The **ComponentOne Studio for ASP.NET Wijmo** installation program will create the following directory: C:\Program Files\ComponentOne\Studio for ASP.NET Wijmo. This directory contains the following subdirectories:

<b>Bin</b>	Contains copies of all binaries (DLLs, Exes) in the ComponentOne Visual Studio ASP.NET Wijmo package.
<b>Wijmo</b>	Contains files (at least a readme.txt) related to the product.

The **ComponentOne Studio for ASP.NET Wijmo Help Setup** program installs integrated Microsoft Help 2.0 and Microsoft Help Viewer help to the C:\Program Files\ComponentOne\Studio for ASP.NET Wijmo directory in the following folders:

<b>H2Help</b>	Contains Microsoft Help 2.0 integrated documentation for all Studio components.
<b>HelpViewer</b>	Contains Microsoft Help Viewer Visual Studio 2010 integrated documentation for all Studio components.

### Samples

Samples for the product are installed in the **ComponentOne Samples** folder by default. The path of the **ComponentOne Samples** directory is slightly different on Windows XP and Windows 7/Vista machines:

**Windows XP path:** C:\Documents and Settings\\My Documents\ComponentOne Samples

**Windows 7/Vista path:** C:\Users\\Documents\ComponentOne Samples

The **ComponentOne Samples** folder contains the following subdirectories:

<b>Common</b>	Contains support and data files that are used by many of the demo programs.
<b>Studio for ASP.NET Wijmo</b>	Contains a readme.txt file and the folders that make up the Control Explorer and other samples.

Samples can be accessed from the **ComponentOne Sample Explorer**. To view samples, on your desktop, click the **Start** button and then click **All Programs | ComponentOne | Studio for ASP.NET Wijmo | Control Explorer**.

## System Requirements

System requirements for **ComponentOne Studio for ASP.NET Wijmo** components include the following:

<b>Operating Systems:</b>	Windows Server® 2003 Windows Server 2008 Windows XP SP2 Windows Vista™ Windows 7
<b>Web Server:</b>	Microsoft Internet Information Services (IIS) 6.0 or later
<b>Environments:</b>	.NET Framework 3.0 or later Visual Studio 2008 or later Internet Explorer 6.0 or later Firefox® 2.0 or later Safari® 2.0 or later

## Uninstalling Studio for ASP.NET Wijmo

To uninstall **Studio for ASP.NET Wijmo**:

1. Open the **Control Panel** and select the **Add or Remove Programs (Programs and Features in Vista/Windows 7)**.
2. Select **ComponentOne Studio for ASP.NET Wijmo** and click the **Remove** button.
3. Click **Yes** to remove the program.

To uninstall **Studio for ASP.NET Wijmo** integrated help:

1. Open the Control Panel and select **Add or Remove Programs** (Programs and Features in Windows 7/Vista).
1. Select **ComponentOne Studio for ASP.NET Wijmo Help** and click the **Remove** button.
2. Click **Yes** to remove the integrated help.

## Deploying your Application in a Medium Trust Environment

Depending on your hosting choice, you may need to deploy your Web site or application in a medium trust environment. Often in a shared hosting environment, medium trust is required. In a medium trust environment several permissions are unavailable or limited, including OleDbPermission, ReflectionPermission, and FileIOPermission. You can configure your Web.config file to enable these permissions.

**Note:** ComponentOne controls will not work in an environment where reflection is not allowed.

ComponentOne ASP.NET Wijmo controls include the `AllowPartiallyTrustedCallers()` assembly attribute and will work under the medium trust level with some changes to the `Web.config` file. Since this requires some control over the `Web.config` file, please check with your particular host to determine if they can provide the rights to override these security settings.

### **Modifying or Editing the Config File**

In order to add permissions, you can edit the existing `web_mediumtrust.config` file or create a custom policy file based on the medium trust policy. If you modify the existing `web_mediumtrust.config` file, all Web applications will have the same permissions with the permissions you have added. If you want applications to have different permissions, you can instead create a custom policy based on medium trust.

#### **Edit the Config File**

In order to add permissions, you can edit the existing `web_mediumtrust.config` file. To edit the existing `web_mediumtrust.config` file, complete the following steps:

1. Locate the medium trust policy file `web_mediumtrust.config` located by default in the `%windir%\Microsoft.NET\Framework\{Version}\CONFIG` directory.
2. Open the `web_mediumtrust.config` file.
3. Add the permissions that you want to grant. For examples, see [Adding Permissions](#) (page 4).

#### **Create a Custom Policy Based on Medium Trust**

In order to add permissions, you can create a custom policy file based on the medium trust policy. To create a custom policy file, complete the following steps:

1. Locate the medium trust policy file `web_mediumtrust.config` located by default in the `%windir%\Microsoft.NET\Framework\{Version}\CONFIG` directory.
2. Copy the `web_mediumtrust.config` file and create a new policy file in the same directory.  
Give the new a name that indicates that it is your variation of medium trust; for example, `AllowReflection_Web_MediumTrust.config`.
3. Add the permissions that you want to grant. For examples, see [Adding Permissions](#) (page 4).
4. Enable the custom policy file on your application by modifying the following lines in your `web.config` file under the `<system.web>` node:

```
<system.web>
<trust level="CustomMedium" originUrl="" />

  <securityPolicy>
    <trustLevel name="CustomMedium"
policyFile="AllowReflection_Web_MediumTrust.config" />
  </securityPolicy>
  ...
</system.web>
```

**Note:** Your host may not allow trust level overrides. Please check with your host to see if you have these rights.

### **Allowing Deserialization**

To allow the deserialization of the license added to `App_Licenses.dll` by the Microsoft IDE, you should add the `SerializationFormatter` flag to security permission to the `Web.config` file. Complete the steps in the [Modifying or Editing the Config File](#) (page 3) topic to create or modify a policy file before completing the following.

Add the `SerializationFormatter` flag to the `<IPermission class="SecurityPermission">` tag so that it appears similar to the following:

```
<NamedPermissionSets>
  <PermissionSet
    class="NamedPermissionSet"
    version="1"
    Name="ASP.Net">
    <IPermission
      class="SecurityPermission"
      version="1"
      Flags="Assertion, Execution, ControlThread,
ControlPrincipal, RemotingConfiguration, SerializationFormatter"/>
    ...
  </PermissionSet>
</NamedPermissionSets>
```

## Adding Permissions

You can add permission, including `ReflectionPermission`, `OleDbPermission`, and `FileIOPermission`, to the web.config file. Note that `ComponentOne` controls will not work in an environment where reflection is not allowed. Complete the steps in the [Modifying or Editing the Config File](#) (page 3) topic to create or modify a policy file before completing the following.

### ReflectionPermission

By default `ReflectionPermission` is not available in a medium trust environment. `ComponentOne ASP.NET Wijmo` controls require reflection permission because `LicenseManager.Validate()` causes a link demand for full trust.

To add reflection permission, complete the following:

1. Open the `web_mediumtrust.config` file or a file created based on the `web_mediumtrust.config` file.
2. Add the following `<SecurityClass>` tag after the `<SecurityClasses>` tag so that it appears similar to the following:

```
<SecurityClasses>
  <SecurityClass Name="ReflectionPermission"
Description="System.Security.Permissions.ReflectionPermission,
mscorlib, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089"/>
  ...
</SecurityClasses>
```

3. Add the following `<IPermission>` tag after the `<NamedPermissionSets>` tag so it appears similar to the following:

```
<NamedPermissionSets>
  <PermissionSet class="NamedPermissionSet" version="1"
Name="ASP.Net">
  <IPermission
    class="ReflectionPermission"
    version="1"
    Flags="ReflectionEmit, MemberAccess" />
  ...
  </PermissionSet>
</NamedPermissionSets>
```

4. Save and close the `web_mediumtrust.config` file.

### OleDbPermission

By default OleDbPermission is not available in a medium trust environment. This means you cannot use the ADO.NET managed OLE DB data provider to access databases. If you wish to use the ADO.NET managed OLE DB data provider to access databases, you must modify the web\_mediumtrust.config file.

To add OleDbPermission, complete the following steps:

1. Open the web\_mediumtrust.config file or a file created based on the web\_mediumtrust.config file.
2. Add the following `<SecurityClass>` tag after the `<SecurityClasses>` tag so that it appears similar to the following:

```
<SecurityClasses>
  <SecurityClass Name="OleDbPermission"
  Description="System.Data.OleDb.OleDbPermission, System.Data,
  Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
  ...
</SecurityClasses>
```

3. Add the following `<IPermission>` tag after the `<NamedPermissionSets>` tag so it appears similar to the following:

```
<NamedPermissionSets>
  <PermissionSet class="NamedPermissionSet" version="1"
  Name="ASP.Net">
    <IPermission class="OleDbPermission" version="1"
  Unrestricted="true"/>
    ...
  </PermissionSet>
</NamedPermissionSets>
```

4. Save and close the web\_mediumtrust.config file.

### FileIOPermission

By default, FileIOPermission is not available in a medium trust environment. This means no file access is permitted outside of the application's virtual directory hierarchy. If you wish to allow additional file permissions, you must modify the web\_mediumtrust.config file.

To modify FileIOPermission to allow read access to a specific directory outside of the application's virtual directory hierarchy, complete the following steps:

1. Open the web\_mediumtrust.config file or a file created based on the web\_mediumtrust.config file.
2. Add the following `<SecurityClass>` tag after the `<SecurityClasses>` tag so that it appears similar to the following:

```
<SecurityClasses>
  <SecurityClass Name="FileIOPermission"
  Description="System.Security.Permissions.FileIOPermission, mscorlib,
  Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
  ...
</SecurityClasses>
```

3. Add the following `<IPermission>` tag after the `<NamedPermissionSets>` tag so it appears similar to the following:

```
<NamedPermissionSets>
  <PermissionSet class="NamedPermissionSet" version="1"
  Name="ASP.Net">
    ...
    <IPermission class="FileIOPermission" version="1"
  Read="C:\SomeDir;$AppDir$" Write="$AppDir$" Append="$AppDir$"
  PathDiscovery="$AppDir$" />
    ...
  </PermissionSet>
```

4. Save and close the web\_mediumtrust.config file.

## End-User License Agreement

All of the ComponentOne licensing information, including the ComponentOne end-user license agreements, frequently asked licensing questions, and the ComponentOne licensing model, is available online at <http://www.componentone.com/SuperPages/Licensing/>.

## Licensing FAQs

This section describes the main technical aspects of licensing. It may help the user to understand and resolve licensing problems he may experience when using ComponentOne .NET and ASP.NET products.

### What is Licensing?

Licensing is a mechanism used to protect intellectual property by ensuring that users are authorized to use software products.

Licensing is not only used to prevent illegal distribution of software products. Many software vendors, including ComponentOne, use licensing to allow potential users to test products before they decide to purchase them.

Without licensing, this type of distribution would not be practical for the vendor or convenient for the user. Vendors would either have to distribute evaluation software with limited functionality, or shift the burden of managing software licenses to customers, who could easily forget that the software being used is an evaluation version and has not been purchased.

### How does Licensing Work?

ComponentOne uses a licensing model based on the standard set by Microsoft, which works with all types of components.

**Note:** The **Compact Framework** components use a slightly different mechanism for run-time licensing than the other ComponentOne components due to platform differences.

When a user decides to purchase a product, he receives an installation program and a Serial Number. During the installation process, the user is prompted for the serial number that is saved on the system. (Users can also enter the serial number by clicking the **License** button on the **About Box** of any ComponentOne product, if available, or by rerunning the installation and entering the serial number in the licensing dialog box.)

When a licensed component is added to a form or Web page, Visual Studio obtains version and licensing information from the newly created component. When queried by Visual Studio, the component looks for licensing information stored in the system and generates a run-time license and version information, which Visual Studio saves in the following two files:

- An assembly resource file which contains the actual run-time license
- A "licenses.licx" file that contains the licensed component strong name and version information

These files are automatically added to the project.

In WinForms and ASP.NET 1.x applications, the run-time license is stored as an embedded resource in the assembly hosting the component or control by Visual Studio. In ASP.NET 2.x applications, the run-time license may also be stored as an embedded resource in the App\_Licenses.dll assembly, which is used to store all run-time licenses for all components directly hosted by WebForms in the application. Thus, the App\_licenses.dll must always be deployed with the application.

The licenses.licx file is a simple text file that contains strong names and version information for each of the licensed components used in the application. Whenever Visual Studio is called upon to rebuild the application

resources, this file is read and used as a list of components to query for run-time licenses to be embedded in the appropriate assembly resource. Note that editing or adding an appropriate line to this file can force Visual Studio to add run-time licenses of other controls as well.

Note that the licenses.licx file is usually not shown in the Solution Explorer; it appears if you press the **Show All Files** button in the Solution Explorer's Toolbox, or from Visual Studio's main menu, select **Show All Files** on the **Project** menu.

Later, when the component is created at run time, it obtains the run-time license from the appropriate assembly resource that was created at design time and can decide whether to simply accept the run-time license, to throw an exception and fail altogether, or to display some information reminding the user that the software has not been licensed.

All ComponentOne products are designed to display licensing information if the product is not licensed. None will throw licensing exceptions and prevent applications from running.

## Common Scenarios

The following topics describe some of the licensing scenarios you may encounter.

### *Creating components at design time*

This is the most common scenario and also the simplest: the user adds one or more controls to the form, the licensing information is stored in the licenses.licx file, and the component works.

Note that the mechanism is exactly the same for Windows Forms and Web Forms (ASP.NET) projects.

### *Creating components at run time*

This is also a fairly common scenario. You do not need an instance of the component on the form, but would like to create one or more instances at run time.

In this case, the project will not contain a licenses.licx file (or the file will not contain an appropriate run-time license for the component) and therefore licensing will fail.

To fix this problem, add an instance of the component to a form in the project. This will create the licenses.licx file and things will then work as expected. (The component can be removed from the form after the licenses.licx file has been created).

Adding an instance of the component to a form, then removing that component, is just a simple way of adding a line with the component strong name to the licenses.licx file. If desired, you can do this manually using notepad or Visual Studio itself by opening the file and adding the text. When Visual Studio recreates the application resources, the component will be queried and its run-time license added to the appropriate assembly resource.

### *Inheriting from licensed components*

If a component that inherits from a licensed component is created, the licensing information to be stored in the form is still needed. This can be done in two ways:

- Add a LicenseProvider attribute to the component.

This will mark the derived component class as licensed. When the component is added to a form, Visual Studio will create and manage the licenses.licx file, and the base class will handle the licensing process as usual. No additional work is needed. For example:

```
[LicenseProvider(typeof(LicenseProvider))]  
class MyGrid: C1.Win.C1FlexGrid.C1FlexGrid  
{  
    // ...  
}
```

- Add an instance of the base component to the form.

This will embed the licensing information into the licenses.licx file as in the previous scenario, and the base component will find it and use it. As before, the extra instance can be deleted after the licenses.licx file has been created.

Please note, that C1 licensing will not accept a run-time license for a derived control if the run-time license is embedded in the same assembly as the derived class definition, and the assembly is a DLL. This restriction is necessary to prevent a derived control class assembly from being used in other applications without a design-time license. If you create such an assembly, you will need to take one of the actions previously described create a component at run time.

### ***Using licensed components in console applications***

When building console applications, there are no forms to add components to, and therefore Visual Studio won't create a licenses.licx file.

In these cases, create a temporary Windows Forms application and add all the desired licensed components to a form. Then close the Windows Forms application and copy the licenses.licx file into the console application project.

Make sure the licenses.licx file is configured as an embedded resource. To do this, right-click the licenses.licx file in the Solution Explorer window and select **Properties**. In the Properties window, set the **Build Action** property to **Embedded Resource**.

### ***Using licensed components in Visual C++ applications***

There is an issue in VC++ 2003 where the licenses.licx is ignored during the build process; therefore, the licensing information is not included in VC++ applications.

To fix this problem, extra steps must be taken to compile the licensing resources and link them to the project. Note the following:

1. Build the C++ project as usual. This should create an .exe file and also a licenses.licx file with licensing information in it.
2. Copy the licenses.licx file from the app directory to the target folder (Debug or Release).
3. Copy the C1Lc.exe utility and the licensed dlls to the target folder. (Don't use the standard lc.exe, it has bugs.)
4. Use C1Lc.exe to compile the licenses.licx file. The command line should look like this:

```
c1lc /target:MyApp.exe /complist:licenses.licx  
/i:C1.Win.C1FlexGrid.dll
```
5. Link the licenses into the project. To do this, go back to Visual Studio, right-click the project, select properties, and go to the Linker/Command Line option. Enter the following:

```
/ASSEMBLYRESOURCE:Debug\MyApp.exe.licenses
```
6. Rebuild the executable to include the licensing information in the application.

### ***Using licensed components with automated testing products***

Automated testing products that load assemblies dynamically may cause them to display license dialog boxes. This is the expected behavior since the test application typically does not contain the necessary licensing information, and there is no easy way to add it.

This can be avoided by adding the string "C1CheckForDesignLicenseAtRuntime" to the AssemblyConfiguration attribute of the assembly that contains or derives from ComponentOne controls. This attribute value directs the ComponentOne controls to use design-time licenses at run time.

For example:

```
#if AUTOMATED_TESTING  
    [AssemblyConfiguration("C1CheckForDesignLicenseAtRuntime")]  
#endif
```

```
public class MyDerivedControl : C1LicensedControl
{
    // ...
}
```

Note that the AssemblyConfiguration string may contain additional text before or after the given string, so the AssemblyConfiguration attribute can be used for other purposes as well. For example:

```
[AssemblyConfiguration("C1CheckForDesignLicenseAtRuntime,BetaVersion")]
```

THIS METHOD SHOULD ONLY BE USED UNDER THE SCENARIO DESCRIBED. It requires a design-time license to be installed on the testing machine. Distributing or installing the license on other computers is a violation of the EULA.

## Troubleshooting

We try very hard to make the licensing mechanism as unobtrusive as possible, but problems may occur for a number of reasons.

Below is a description of the most common problems and their solutions.

### ***I have a licensed version of a ComponentOne product but I still get the splash screen when I run my project.***

If this happens, there may be a problem with the licenses.licx file in the project. It may not exist, it may contain incorrect information, or it may not be configured correctly.

First, try a full rebuild (**Rebuild All** from the Visual Studio **Build** menu). This will usually rebuild the correct licensing resources.

#### **If that fails follow these steps:**

1. Open the affected project.
2. Select an instance of the updated component.
3. In the Visual Studio Properties window, change any property. It does not matter which property you change; you can change it back to the previous value.
4. Rebuild the project using the **Rebuild All** option (not just **Rebuild**) and run the solution.

#### **Alternative 1: Follow these steps:**

1. Open a new Visual Studio.NET project.
2. Add the updated component to the form.
3. Compile and run the new project.
4. Open the licenses.licx file in the new project.
5. Copy the line that starts with the namespace of the updated component (for example, C1.Win.C1Report) and ends with a public key token.
6. Open the existing, incorrectly licensed project.
7. Open the licenses.licx file in the new project.
5. Paste the line from step 5 into this file (replace the old licensing information with the new).
6. Rebuild the project using the **Rebuild All** option (not just **Rebuild**) and run the solution.

#### **Alternative 2: Follow these steps:**

1. Open the affected project.
2. Delete the licenses.licx file from the project.
3. Add a new instance of the updated component to the form.

4. Rebuild and run the solution. The nag screen should not appear.
5. Remove the newly added component from the form.

Try each of these options multiple times, if necessary. If that still does not help, are you creating any of the controls in code rather than design-time? If so, you must add an entry for the control in the licenses.licx file (see <http://helpcentral.componentone.com/PrintableView.aspx?ID=1886> for more information). Also if this is a website, as opposed to an ASP.NET web application, please try right-clicking the licenses.licx file and selecting "Build Runtime Licenses" from the context menu.

### ***I have a licensed version of a ComponentOne product on my Web server but the components still behave as unlicensed.***

There is no need to install any licenses on machines used as servers and not used for development.

The components must be licensed on the development machine, therefore the licensing information will be saved into the executable (.exe or .dll) when the project is built. After that, the application can be deployed on any machine, including Web servers.

For ASP.NET 2.x applications, be sure that the App\_Licenses.dll assembly created during development of the application is deployed to the bin application bin directory on the Web server.

If your ASP.NET application uses WinForms user controls with constituent licensed controls, the run-time license is embedded in the WinForms user control assembly. In this case, you must be sure to rebuild and update the user control whenever the licensed embedded controls are updated.

### ***I downloaded a new build of a component that I have purchased, and now I'm getting the splash screen when I build my projects.***

Make sure that the serial number is still valid. If you licensed the component over a year ago, your subscription may have expired. In this case, you have two options:

#### **Option 1 – Renew your subscription to get a new serial number.**

If you choose this option, you will receive a new serial number that you can use to license the new components (from the installation utility or directly from the **About Box**).

The new subscription will entitle you to a full year of upgrades and to download the latest maintenance builds directly from <http://prerelease.componentone.com/>.

#### **Option 2 – Continue to use the components you have.**

Subscriptions expire, products do not. You can continue to use the components you received or downloaded while your subscription was valid.

## Technical Support

ComponentOne offers various support options. For a complete list and a description of each, visit the ComponentOne Web site at <http://www.componentone.com/SuperProducts/SupportServices/>.

Some methods for obtaining technical support include:

- **[Online Resources](#)**  
ComponentOne provides customers with a comprehensive set of technical resources in the form of FAQs, samples and videos, Version Release History, searchable Knowledge base, searchable Online Help and more. We recommend this as the first place to look for answers to your technical questions.
- **Online Support via our Incident Submission Form**  
This online support service provides you with direct access to our Technical Support staff via an [online incident submission form](#). When you submit an incident, you'll immediately receive a response via e-mail confirming that you've successfully created an incident. This email will provide you with an Issue Reference ID

and will provide you with a set of possible answers to your question from our Knowledgebase. You will receive a response from one of the ComponentOne staff members via e-mail in 2 business days or less.

- **Product Forums**  
ComponentOne's [product forums](#) are available for users to share information, tips, and techniques regarding ComponentOne products. ComponentOne developers will be available on the forums to share insider tips and technique and answer users' questions. Please note that a ComponentOne User Account is required to participate in the ComponentOne Product Forums.
- **Installation Issues**  
Registered users can obtain help with problems installing ComponentOne products. Contact technical support by using the [online incident submission form](#) or by phone (412.681.4738). Please note that this does not include issues related to distributing a product to end-users in an application.
- **Documentation**  
Microsoft integrated ComponentOne documentation can be installed with each of our products, and documentation is also available online. If you have suggestions on how we can improve our documentation, please email the [Documentation team](#). Please note that e-mail sent to the [Documentation team](#) is for documentation feedback only. [Technical Support](#) and [Sales](#) issues should be sent directly to their respective departments.

**Note:** You must create a ComponentOne Account and register your product with a valid serial number to obtain support using some of the above methods.

## Redistributable Files

**ComponentOne Studio for ASP.NET Wijmo** is developed and published by ComponentOne LLC. You may use it to develop applications in conjunction with Microsoft Visual Studio or any other programming environment that enables the user to use and integrate the control(s). You may also distribute, free of royalties, the following Redistributable Files with any such application you develop to the extent that they are used separately on a single CPU on the client/workstation side of the network:

- C1.Web.Wijmo.Controls.3.dll
- C1.Web.Wijmo.Controls.Design.3.dll
- C1.Web.Wijmo.Controls.4.dll
- C1.Web.Wijmo.Controls.Design.4.dll
- C1.Web.Wijmo.Extenders.3.dll
- C1.Web.Wijmo.Extenders.4.dll
- C1.C1Report.2.dll
- C1.C1Report.4.dll

Site licenses are available for groups of multiple developers. Please contact [Sales@ComponentOne.com](mailto:Sales@ComponentOne.com) for details.

## About This Documentation

### Acknowledgements

*Microsoft, Windows, Windows Vista, Visual Studio, and Microsoft Expression are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.*

Firefox is a registered trademark of the Mozilla Foundation.

Safari is a registered trademark of Apple Inc.

### ComponentOne

If you have any suggestions or ideas for new features or controls, please call us or write:

*Corporate Headquarters*

**ComponentOne LLC**

201 South Highland Avenue

3<sup>rd</sup> Floor

Pittsburgh, PA 15206 • USA

412.681.4343

412.681.4384 (Fax)

<http://www.componentone.com>

**ComponentOne Doc-To-Help**

This documentation was produced using [ComponentOne Doc-To-Help® Enterprise](#).

## Namespaces

Namespaces organize the objects defined in an assembly. Assemblies can contain multiple namespaces, which can in turn contain other namespaces. Namespaces prevent ambiguity and simplify references when using large groups of objects such as class libraries.

The general namespace for ComponentOne Web products is **C1.Web**. The following code fragment shows how to declare a **C1ComboBox** using the fully qualified name for this class:

- Visual Basic

```
Dim combobox As C1.Web.Wijmo.Controls.C1ComboBox
```

- C#

```
C1.Web.Wijmo.Controls.C1ComboBox combobox;
```

Namespaces address a problem sometimes known as *namespace pollution*, in which the developer of a class library is hampered by the use of similar names in another library. These conflicts with existing components are sometimes called *name collisions*.

Fully qualified names are object references that are prefixed with the name of the namespace where the object is defined. You can use objects defined in other projects if you create a reference to the class (by choosing Add Reference from the Project menu) and then use the fully qualified name for the object in your code.

Fully qualified names prevent naming conflicts because the compiler can always determine which object is being used. However, the names themselves can get long and cumbersome. To get around this, you can use the Imports statement (**using** in C#) to define an alias — an abbreviated name you can use in place of a fully qualified name. For example, the following code snippet creates aliases for two fully qualified names, and uses these aliases to define two objects:

- Visual Basic

```
Imports C1ComboBox = C1.Web.UI.Controls.C1Input.C1InputMask
Imports MyComboBox = MyProject.Objects.C1Input.C1InputMask

Dim wm1 As C1ComboBox
Dim wm2 As MyComboBox
```

- C#

```
using C1ComboBox = C1.Web.UI.Controls.C1ComboBox;
using MyComboBox = MyProject.Objects.C1ComboBox;

C1ComboBox wm1;
MyComboBox wm2;
```

If you use the **Imports** statement without an alias, you can use all the names in that namespace without qualification provided they are unique to the project.

## Creating an ASP.NET Project

**ComponentOne Studio for ASP.NET Wijmo** requires Visual Studio 2008 or later and .NET Framework 3.0 or later for your Web applications. **Studio for ASP.NET Wijmo** does not require AJAX extensions; however, you can install them if you want to use AJAX in your project. See the following table for more details on installing AJAX extensions.

Visual Studio 2010	You can build Ajax-enabled ASP.NET projects by downloading the AJAX Control Toolkit from <a href="#">CodePlex</a> and dragging-and-dropping the controls from the Visual Studio Toolbox onto an ASP.NET Web Forms page.
Visual Studio 2008, .NET Framework 3.5	You can easily create an AJAX-enabled ASP.NET project without installing separate add-ins because the framework has a built-in AJAX library and controls.
Visual Studio 2008, .NET Framework 3.0	You must install the ASP.NET AJAX Extensions 1.0, which can be found at <a href="http://www.asp.net/ajax/downloads/archive/">http://www.asp.net/ajax/downloads/archive/</a> . Then you can create an AJAX 1.0-Enabled ASP.NET 2.0 Web site or application.

The following topics explain how to create projects in Visual Studio 2010 and 2008.

- **Creating a Web Site/Application Project in Visual Studio 2010**

To create a new Web site/application project in Visual Studio 2010, complete the following steps.

1. If you are creating an AJAX project, download the AJAX Control Toolkit from [CodePlex](#) and extract the files.
2. From the **File** menu, select **New | Web Site/Project**. The New Web Site/New Project dialog box opens.
3. Select a .NET Framework in the upper right corner. Note that if you choose .NET Framework 3.0, you must install the [extensions](#) first.
4. Under **Project Types**, choose either **Visual Basic** or **Visual C#** and then select **Web**. Note that one of these options may be located under **Other Languages**.
5. Select a language, and in the list of templates, select **ASP.NET Web Site/Application**.
6. Specify a location and then click **OK**.

**Note:** The Web server must have IIS version 6 or later and the .NET Framework installed on it. If you have IIS on your computer, you can specify `http://localhost` for the server.

A new Web project is created at the root of the Web server you specified.

7. If you are creating an AJAX project, right-click the Toolbox (create a new tab if you like), select **Choose Items** and browse to find the **AjaxControlToolkit.dll**. You can begin dragging toolkit controls to your page. Note that if you do not see the toolkit controls in the Toolbox, make sure you selected the .NET Framework that corresponds with the version of the toolkit you downloaded.

- **Creating a Web Site/Application Project in Visual Studio 2008**

To create a Web site/application project in Visual Studio 2008, complete the following steps:

1. From the **File** menu, select **New | Web Site/Project**. The New Web Site/New Project dialog box opens.

2. Select .NET Framework 3.5 or 3.0 in the upper right corner. Note that if you choose .NET Framework 3.0, you must install the [extensions](#) first.
3. Select a language, and in the list of templates, select **ASP.NET Web Site/Application** or **AJAX 1.0-Enabled ASP.NET 2.0 Web Site/Application**.
4. Specify a location and then click **OK**.

**Note:** The Web server must have IIS version 6 or later and the .NET Framework installed on it. If you have IIS on your computer, you can specify http://localhost for the server.

A new Web project is created at the root of the Web server you specified.

## Adding the ComboBox for ASP.NET Wijmo Component to a Project

When you open Visual Studio, you will notice a **ComponentOne Studio for ASP.NET Wijmo Projects** tab containing the ComponentOne controls that have automatically been added to the Toolbox.

Note that you can manually add ComponentOne controls to the Toolbox at a later time.

### Manually Adding the Studio for ASP.NET Wijmo controls to the Toolbox

When you install **ComponentOne Studio for ASP.NET Wijmo**, the following **ComboBox for ASP.NET Wijmo** components will appear in the Visual Studio Toolbox customization dialog box:

- C1ComboBox

To manually add the Studio for ASP.NET Wijmo controls to the Visual Studio Toolbox:

1. Open the Visual Studio IDE (Microsoft Development Environment). Make sure the Toolbox is visible (select **Toolbox** in the **View** menu if necessary) and right-click it to open the context menu.
2. To make the Studio for ASP.NET Wijmo components appear on their own tab in the Toolbox, select **Add Tab** from the context menu and type in the tab name, Studio for ASP.NET Wijmo, for example.
3. Right-click the tab where the component is to appear and select **Choose Items** from the context menu. The **Choose Toolbox Items** dialog box opens.
4. In the dialog box, select the **.NET Framework Components** tab. Sort the list by Namespace (click the **Namespace** column header) and check the check boxes for all components belonging to namespace C1.Web.Wijmo.Controls.C1Input. Note that there may be more than one component for each namespace.
5. Click **OK** to close the dialog box. The controls are added to the Visual Studio Toolbox.

### Adding Studio for ASP.NET Wijmo Controls to the Form

To add **Studio for ASP.NET Wijmo** controls to a form:

1. Add them to the Visual Studio Toolbox.
2. Double-click each control or drag it onto your form.

### Adding a Reference to the Assembly

To add a reference to the C1.Web.Wijmo.Controls.3 or C1.Web.Wijmo.Controls.4 assembly:

1. Select the **Add Reference** option from the **Website** menu of your Web Site project or from the **Project** menu of your Web Application project.
2. Select the most recent version of the **ComponentOne Studio for ASP.NET Wijmo** assembly from the list on the **NET** tab or browse to find the C1.Web.Wijmo.Controls.3.dll or C1.Web.Wijmo.Controls.4.dll file and click **OK**.

3. Select the **Form1.vb** tab or go to **View | Code** to open the Code Editor. At the top of the file, add the following **Imports** directive (**using** in C#):

```
Imports Cl.Web.Wijmo.Controls
```

**Note:** This makes the objects defined in the **C1.Web.Wijmo.Controls.3(4)** assembly visible to the project. See [Namespaces](#) (page 12) for more information.



# Key Features

**ComponentOne ComboBox for ASP.NET Wijmo** provides the following unique key features:

- **Auto-searchable Drop-down list**  
Locate items quickly by typing the first few characters. ComboBox will automatically search the list and select the items for you as you type.
- **Load on Demand**  
Populate items dynamically from the server using AJAX. This helps to keep page sizes small and manageable.
- **Item Selection**  
The C1ComboBox control provides single and multiple selection modes allowing end-users to select one or several items in the drop-down list.
- **Resizable Drop-down List**  
The C1ComboBox control provides single and multiple selection modes allowing end-users to select one or several items in the drop-down list.
- **Theming**  
With just a click of the SmartTag, change the bar chart's look by selecting one of the 5 premium themes (Midnight, Aristo, Rocket, Cobalt, and Sterling). Optionally, use ThemeRoller from jQuery UI to create a customized theme.
- **CSS Support**  
Use a cascading style sheet (CSS) style to define custom skins. CSS support allows you to match the accordion to your organization's standards.

## ComboBox for ASP.NET Wijmo Quick Start

The C1ComboBox Quick Start describes how to get started with the ASP.NET control, **C1ComboBox**. In this quick start, you will create an ASP.NET application containing one **C1ComboBox** control and three **C1ComboBoxItems**. Declare one Label control in an .aspx file. Then create an event handler for the **SelectedItem** event which displays the value of the selected combobox item.

### Step 1 of 4: Creating an Application

In this topic you will create an ASP.NET web site and add a **C1ComboBox** control to the Default.aspx page.

1. Begin by creating an ASP.NET Web application. Note that if using Visual Studio 2008, you must add a **ScriptManager** control to the form. If using Visual Studio 2005, the **ScriptManager** control is automatically added to the form.
1. Add the following references to your project:
  - C1.Web.Wijmo.Controls.4.dll
  - C1.Web.Wijmo.Controls.Design.4.dll

- C1.C1Report.4.dll

2. Add the controls to the Toolbox.

Open the Visual Studio IDE (Microsoft Development Environment). Make sure the Toolbox is visible (select **Toolbox** in the **View** menu if necessary) and right-click it to open the context menu.

To make the Studio for ASP.NET Wijmo components appear on their own tab in the Toolbox, select **Add Tab** from the context menu and type in the tab name, Studio for ASP.NET Wijmo, for example.

Right-click the tab where the component is to appear and select **Choose Items** from the context menu. The **Choose Toolbox Items** dialog box opens.

In the dialog box, select the **.NET Framework Components** tab. Sort the list by Namespace (click the **Namespace** column header) and check the check boxes for all components belonging to namespace C1.Web.Wijmo.Controls.C1ComboBox.

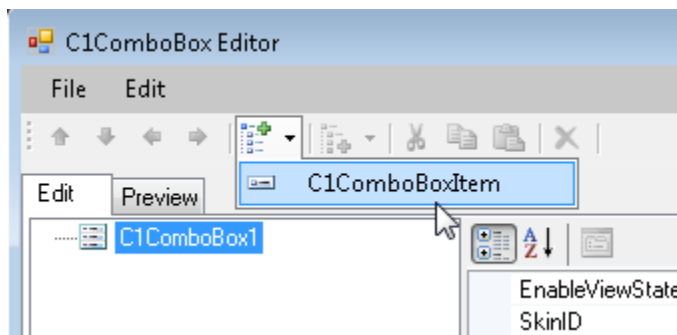
Click **OK** to close the dialog box. The controls are added to the Visual Studio Toolbox.

3. Select the Design tab and remove the default content from the Default.aspx page.
4. While in Design view, navigate to the Visual Studio Toolbox and double-click the **C1ComboBox** control to add a combobox to the main content of your page.

## Step 2 of 4: Adding Items to C1ComboBox

In this step, you will use the **C1ComboBox Editor** to add items to a **C1ComboBox** control that will appear at run time when you click on its dropdown arrow.

1. Click the **C1ComboBox** smart tag and select **Edit Items** from the **C1ComboBox Tasks** menu. The **C1ComboBox Editor** appears.
2. Select the C1ComboBoxItem from the **Add Child Item** button three times to get three C1ComboBoxItems.



3. Change the C1ComboItem1 Text property to “Coach” and Value property to “Coach Cabin”, C1ComboItem2 Text property to “Business” and Value property to “Business Cabin” and C1ComboItem3 Text property to “First” and Value property to “First Cabin”.
4. Click **OK** to apply the changes to the C1ComboBox control and close the C1ComboBox Editor.

## Step 3 of 4: Create an Event Handler for the Selected Item

In this topic the selectedItem event is triggered when an item in the combobox is selected.

2. Select the Source tab and add the following script for the **C1ComboBox1\_OnClientChanged** function within the HeaderContent tag:

```
<asp:Content ID="HeaderContent" runat="server"
ContentPlaceHolderID="HeadContent">
```

```

<script id="scriptInit" type="text/javascript">
function C1ComboBox1_OnClientChanged(e, data) {
    var val = data.selectedItem.value;
    $('#output').html('I selected the ' + val + ' airfare option.
');
}
</script>

</asp:Content>

```

1. Add a Label to the source page after the C1ComboBox tags.

```

<label id="output">
    Please select fare option.</label>

```

2. Within the C1ComboBox tags assign the onclientchanged="C1ComboBox1\_OnClientChanged" function to the onclientchanged property so it appears like the following:

```

<asp:Content ID="BodyContent" runat="server"
ContentPlaceHolderID="MainContent">
    <wijmo:C1ComboBox ID="C1ComboBox1" runat="server" Width="160px"
onclientchanged="C1ComboBox1_OnClientChanged">

```

## Step 4 of 4: Running the Project

Press **F5** to run the project and view the following:

- Select an item from the list and notice the value of the item selected appears updated in the text below.



Congratulations! You have successfully completed the **ComboBox for ASP.NET Wijmo Quick Start**.

# Wijmo Top Tips

The following tips may help you troubleshoot when working with **Studio for ASP.NET Wijmo**.

### Tip 1: Prevent poor page rendering in quirks mode by editing the meta tag to fix rendering.

If a user's browser is rendering a page in quirks mode, widgets and controls may not appear correctly on the page. This is indicated by a broken page icon in the address bar. In **Compatibility View**, the browser uses an older rendering engine.



Users can set this view that causes the issue. To prevent rendering in quirks mode, you can force the page to render with the latest browser. Add the following meta tag to the header of the page:

```

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

```

# Design-Time Support

C1ComboBox provides customized context menus, smart tags, and a designer that offers rich design-time support and simplifies working with the object model.

The following sections describe how to use C1ComboBox design-time environment to configure the **C1ComboBox** control.

## C1ComboBox Smart Tag

In Visual Studio, the **C1ComboBox** control includes a smart tag. A smart tag represents a short-cut tasks menu that provides the most commonly used properties in **C1ComboBox**.

To access the **C1ComboBox Tasks** menu, click on the smart tag (☒) in the upper-right corner of the **C1ComboBox** control. This will open the **C1ComboBox Tasks** menu.



The **C1ComboBox Tasks** menu operates as follows:

### Choose Data Source

Clicking on the **Choose Data Source** item opens a drop-down list where you can choose an existing data source or select a new data source to bind to.

### Edit Items

Clicking the **Edit Items** link item opens the **C1ComboBox** editor.

### Edit Columns

Clicking the Edit Columns link item opens the **Columns Designer** editor.

### Theme

The **Theme** drop-down box allows you to set the **Theme** property and change the C1ComboBox control's appearance to one of the five predefined themes. By default this is set to the **Aristo** theme. For more information about available visual styles, see [Themes](#) (page 25).

### Use CDN

When the **Use CDN** checkbox is selected it loads the client resources from CDN. This is not selected by default.

CDN path

Displays the url path of the **CDN**.

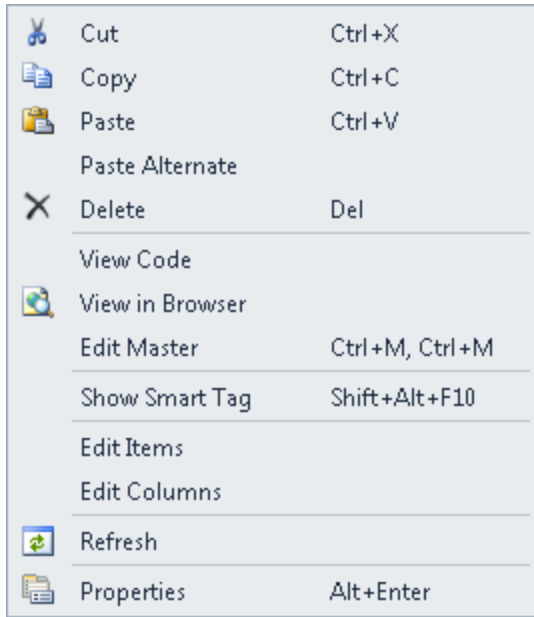
## About

Clicking on the **About** item displays the **About Studio for ASP.NET Wijmo** dialog box, which is helpful in finding the version number of **ComponentOne Studio for ASP.NET Wijmo** and online resources.

## C1ComboBox Context Menu

C1ComboBox has additional commands with each context menu that Visual Studio provides for all .NET and ASP.NET controls.

Right-click anywhere on the list to display the **C1ComboBox** context menu:



The context menu commands operate as follows:

### Edit Items

Selecting the **Edit Items** menu item opens the **C1ComboBox** editor.

### Edit Columns

Selecting the **Edit Columns** menu item opens the **Columns Designer** editor.

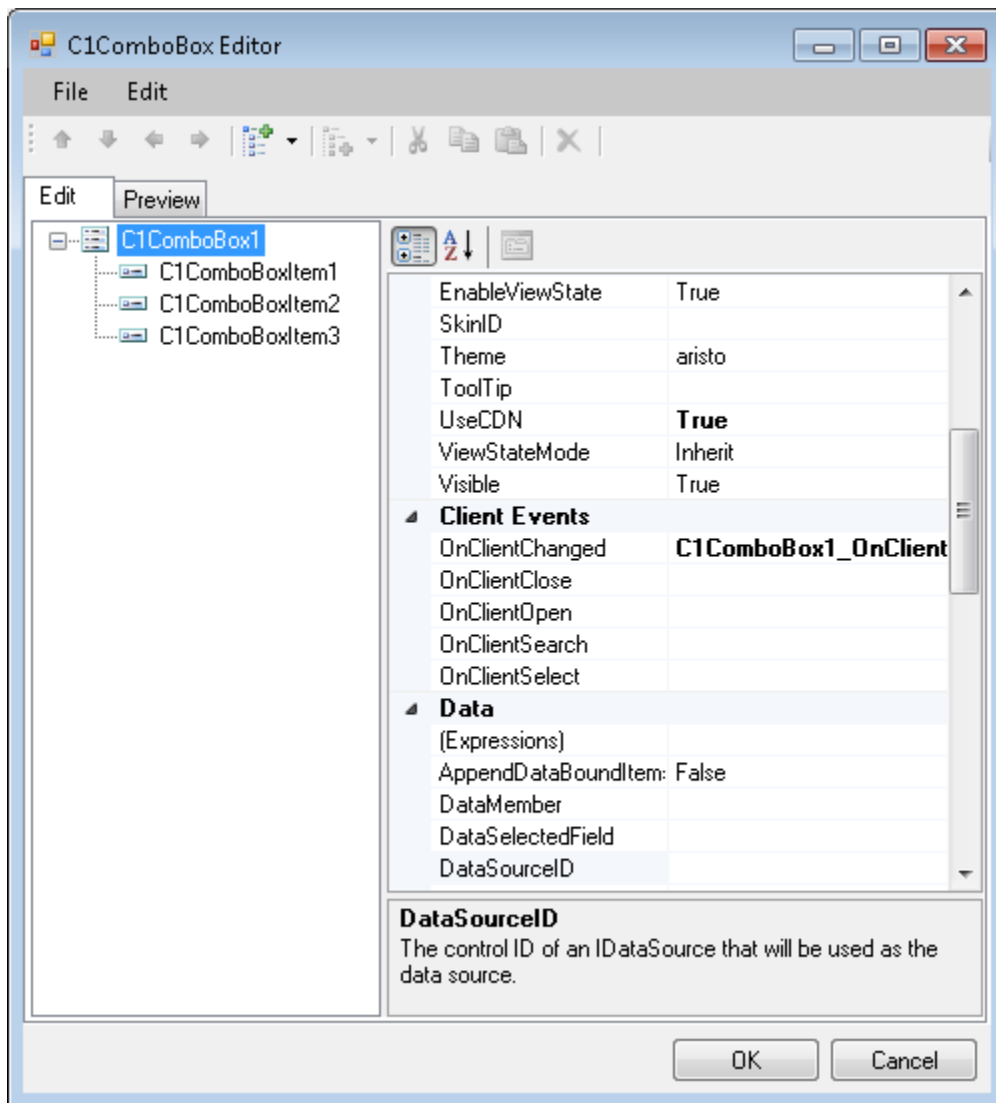
## C1ComboBox Collection Editors

C1ComboBox includes the following two collection editors to add/remove/modify C1CombBoxItems and columns.

- **C1ComboBox Editor**
- **Columns Designer**

### C1ComboBox Editor

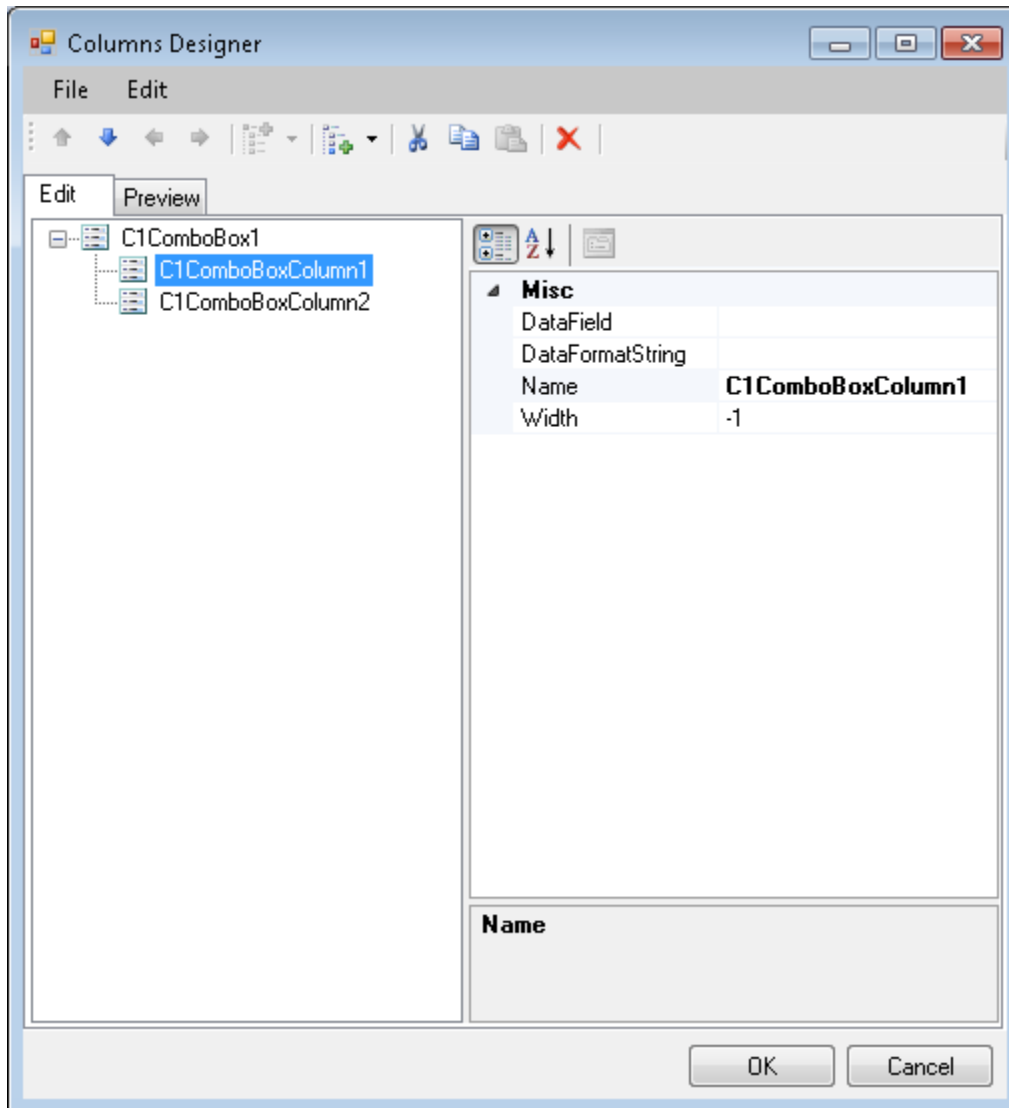
The **C1ComboBox Editor** allows the user to add/remove/modify C1CombBoxItems in the C1CombBox control.



To access the **C1ComboBox Editor**, right-click on the **C1ComboBox** control and select **Edit Items** from its context menu.

## Columns Designer

The **Columns Designer** allows you add, remove, or modify columns in the **C1ComboBox** control.



To access the **Columns Designer**, right-click on the **C1ComboBox** control and select **Columns Designer** from its context menu.

## ComboBox Appearance

The combobox's appearance is controlled by the built-in themes or the CSS styles. The combobox themes effects the appearance of all of the combobox elements such as the combobox items and columns.

C1ComboBox is designed to make customization easy for you. You have endless possibilities in changing its default appearance.

You can apply CSS styles to modify C1ComboBox's elements. C1ComboBox provides several built-in CSS Selectors that appear when you select the drop down arrow next to the **CSSClass** property

# C1ComboBox CSS Selectors

You can style any element of the C1ComboBox using CSS styles to make their appearance truly unique. To make the customization process easier, ComponentOne includes CSS selectors for each of its six built-in themes.

You can apply general CSS properties such as background, text, font, border, outline, margin, padding, list, and table to applicable CSS selectors.

For a list of common individual CSS selectors and grouped CSS selectors, select the **C1ComboBox** control in your project, and view the drop-down list next to the **CssClass** property in the **C1ComboBox Visual Studio Properties** window.

The following table details the common individual CSS selectors and grouped CSS selectors. You can combine the individual CSS selectors as a group to make the CSS selector more specific and strong. The grouped CSS selectors define styles for more than one element of the C1ComboBox.

CSS Selectors	Description
.wijmo-wijcombobox	Applies the style to the wijcombobox control.
.wijmo-wijcombobox-active-prev	Applies the style to the previous active wijcombobox.
.wijmo-wijcombobox-cell	Applies the style to the cell in the wijcombobox control
.wijmo-wijcombobox-input	Applies the style to the input wijcombobox.
.wijmo-wijcombobox-item ui-state-active	Applies the style to the active state unlisted item of the wijcombobox.
.wijmo-wijcombobox-item ui-state-hover	Applies the style to the hover state unlisted item of the wijcombobox.
.wijmo-wijcombobox-label	Applies the style to the label of the wijcombobox.
.wijmo-wijcombobox-list	Applies the style to the list of the wijcombobox.
.wijmo-wijcombobox-loading	Applies the style to the loading wijcombobox.
.wijmo-wijcombobox-multicolumn	Applies the style to the multicolumn wijcombobox.
.wijmo-wijcombobox-row	Applies the style to the wijcombobox row.
.wijmo-wijcombobox-rowheader	Applies the style to the rowheader of the wijcombobox.
.wijmo-wijcombobox-trigger	Applies the style to the triggered wijcombobox.
.wijmo-wijcombobox-ul	Applies the style to the unlisted wijcombobox.

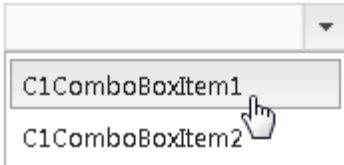
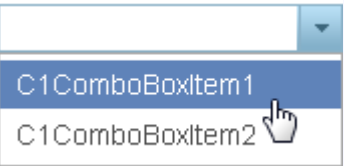
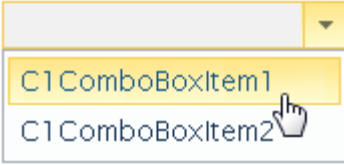
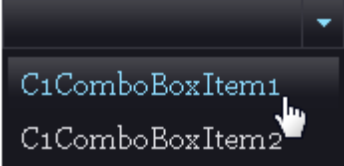

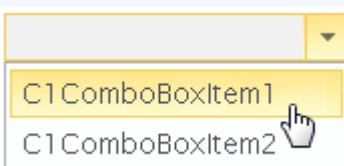
.wijmo-wijcombobox-wrapper

Applies the style to the wrapper  
wijcombobox.

## Themes

C1ComboBox provides five built-in visual styles for the control – **Arctic**, **Aristo**, **Cobalt**, **Midnight**, and **Rocket**, and **Sterling** - that can be easily applied to the control by setting the **Theme** property.

The following table illustrates each of the six built-in themes with the hover style shown.

Theme	Appearance
arctic	
Aristo (default)	
Cobalt	
Midnight	
Rocket	
Sterling	

# Working with the Client-Side ComboBox

**C1ComboBox** client side has a very rich client-side object model since most of its members are identical to the members in the server-side control.

When a **C1ComboBox** control is rendered, an instance of the client-side combobox will be created automatically. This means that you can enjoy the convenience of accessing properties and methods of the **C1ComboBox** control without having to postback to the server.

Using **C1ComboBox**'s client-side code, you can implement many features in your Web page without the need to send information to the Web server which takes time. Thus, using **C1ComboBox**'s client-side object model can increase the efficiency of your Web site.

## Client-Side Events

**C1ComboBox** includes several client-side events that allow you to manipulate the combobox items in the **C1ComboBox** control when an action such as selecting the item, opening the drop-down list, or closing the drop-down list occurs.

Each of the client-side events requires two parameters: the **ID** that identifies the sender **C1ComboBox**.

You can use the sever-side properties, listed in the **Client Side Event** table, to specify the name of the JavaScript function that will respond to a particular client-side event. For example, to assign a JavaScript function called "CloseList" to respond to the closed drop-down list, you would set the **OnClientClose** property to "CloseList".

The following table lists the events that you can use in your client scripts. These properties are defined on the server side, but the actual events or the name you declare for each JavaScript function are defined on the client side.

**Client Side Event table**

<b>Event Server-Side Property Name</b>	<b>Event Name</b>	<b>Description</b>
OnClientChanged	Changed	A function called when the select item is changed
OnClientClose	Close	A function called when the drop-down list is closed.
OnClientOpen	Open	A function called when the drop-down list is opened.
OnClientSearch	Search	A function called before searching the list.
OnClientSelect	Select	A function called when any item in the list is selected.

# ComboBox for ASP.NET Wijmo Samples

Please be advised that this ComponentOne software tool is accompanied by various sample projects and/or demos, which may make use of other ComponentOne development tools included with ComponentOne Studio Enterprise.

## C# Samples

The following pages within the **ControlExplorer** sample installed with **ComponentOne Studio for Wijmo** detail the C1ComboBox control's functionality:

Sample	Description
Animation	This sample demonstrates the different animation effects and how you can manipulate their speed.
DataBinding	Demonstrates how to bind C1ComboBox to an XML or Access data source.
LoadOnDemand	This sample demonstrates the different callback events and how you can populate items for c1combobox without postback their speed.
MultipleColumns	Shows how you can easily create multiple columns in the c1combobox by setting the columns option.
Overview	Displays an editable text box/drop-down list on the aspx page.
Position	This sample shows how you can change the position of the drop-down list for the c1combobox by setting the dropDownListPosition option.
Select	This sample illustrates how the select event is triggered when an item in the combobox is selected.



# ComboBox for ASP.NET Wijmo Task-Based Help

The task-based help assumes that you are familiar with programming in Visual Studio .NET. By following the steps outlined in the Help, you will be able to create projects demonstrating a variety of **ComboBox for ASP.NET Wijmo** features and get a good sense of what **ComboBox for ASP.NET Wijmo** can do.

## Adding a Custom Theme

**ComboBox for ASP.NET Wijmo** provides six built-in themes, but if you prefer to use a different theme, you can choose an existing theme using a CDN URL or create your own custom theme with the jQuery ThemeRoller Web application. .

### Using a CDN URL

1. Click the **C1ComboBox** smart tag to open the Tasks menu.
2. Select **Use CDN**.
3. In the **Theme** property enter a CDN URL to specify the theme; CDN URLs can be found at <http://blog.jqueryui.com/2011/06/jquery-ui-1-8-14/>. In this example, we'll use the sunny theme: <http://ajax.aspnetcdn.com/ajax/jquery.ui/1.8.14/themes/sunny/jquery-ui.css>.

This theme setting is stored in the <appSettings> of the Web.config file. In the Solution Explorer, double-click the Web.config file. Notice the <appSettings> tag contains a WijmoTheme key and value; this is where the CDN URL you added is specified.

4. Run the project and notice the theme is applied to ComboBox.

### Using jQuery ThemeRoller

1. Go to <http://jqueryui.com/themeroller/>.
2. On the **Roll Your Own** tab, change the settings to create a custom theme; you can customize fonts, colors, backgrounds, borders, and more. Or click the Gallery tab and select an existing theme.
3. Click the Download button and then click Download again on the Build Your Download page.
4. Save and unzip the theme .zip file to a folder within your Visual Studio project folder. In this example, we created a **Themes** folder.
5. In the Solution Explorer, select the project name and click the refresh button so the **Themes** folder appears in your project.
6. Click the **C1ComboBox** smart tag to open the **Tasks** menu.
7. Select **Use CDN**.
8. Right-click on the **C1ComboBox** control and navigate to the **Theme** property in the Properties window and enter the path to your custom theme .css, for example, **Themes/css/custom-theme/jquery-ui-1.8.16.custom.css**.

This theme setting is stored in the <appSettings> of the Web.config file. In the Solution Explorer, double-click the Web.config file. Notice the <appSettings> tag contains a WijmoTheme key and value; this is where the custom theme you added is specified.

9. Run the project and notice them theme is applied to **C1ComboBox**.



# Combobox for ASP.NET Wijmo Client-Side Reference

As part of the amazing [ComponentOne Web stack](#), the Wijmo jQuery UI widgets are optimized for client-side Web development and utilize the power of jQuery for superior performance and ease of use.

The ComponentOne Wijmo website at <http://wijmo.com/widgets/> provides everything you need to know about Wijmo widgets, including demos and samples, documentation, theming examples, support and more.

The client-side documentation provides an overview, sample markup, options, events, and methods for each widget. To get started with client-side Web development for **Combobox for ASP.NET Wijmo**, click one of the external links to view our Wijmo wiki documentation. Note that the **Overview** topic for each of the widgets applies mainly to the widget, not to the server-side ASP.NET Wijmo control.

## Combobox

[wijcombobox dependencies](#)

[wijcombobox options](#)

[wijcombobox events](#)

## Using the Wijmo CDN

You can easily load the client-side Wijmo widgets into your web page using a Content Delivery Network (CDN). CDN makes it quick and easy to use external libraries, and deploy them to your users. A CDN is a network of computers around the world that host content. Ideally, if you're in the United States and you access a webpage using a CDN, you'll get your content from a server based in the US. If you're in India or China, and you access the SAME webpage, the content will come from a server a little closer to your location.

When web browsers load content, they commonly will check to see if they already have a copy of the file cached. By using a CDN, you can benefit from this. If a user had previously visited a site using the same CDN, they will already have a cached version of the files on their machine. Your page will load quicker since it doesn't need to re-download your support content.

Wijmo has had CDN support from the very beginning. You can find the CDN page at <http://wijmo.com/downloads/cdn/>. The markup required for loading Wijmo into your page looks similar to this:

```
<!--jQuery References-->
<script src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-1.7.1.min.js"
type="text/javascript"></script>
<script src="http://ajax.aspnetcdn.com/ajax/jquery.ui/1.8.17/jquery-
ui.min.js" type="text/javascript"></script>
<!--Theme-->
<link href="http://cdn.wijmo.com/themes/rocket/jquery-wijmo.css"
rel="stylesheet" type="text/css" title="rocket-jqueryui" />
<!--Wijmo Widgets CSS-->
<link href="http://cdn.wijmo.com/jquery.wijmo-
complete.all.2.0.0.min.css" rel="stylesheet" type="text/css" />
<!--Wijmo Widgets JavaScript-->
<script src="http://cdn.wijmo.com/jquery.wijmo-open.all.2.0.0.min.js"
type="text/javascript"></script>
<script src="http://cdn.wijmo.com/jquery.wijmo-
complete.all.2.0.0.min.js" type="text/javascript"></script>
```

In this markup, you'll notice that some of the .js files are labeled as \*.min.js. These files have been minified - in other words, all unnecessary characters have been removed - to make the pages load faster. You will also notice that there are

no references to individual .js files. The JavaScript for all widgets, CSS, and jQuery references have been combined into one file, respectively, such as wijmo-complete.2.0.0.min.js. If you want to link to individual .js files, see the **Dependencies** topic for each widget.

With the **ComponentOne Studio for ASP.NET Wijmo** controls, you can click the **Use CDN** checkbox in the control's **Tasks** menu and specify the **CDN path** if you want to access the client-side widgets.

C1ComboBox Tasks	
Choose Data Source:	<input type="text" value="(None)"/>
<a href="#">Edit Items</a>	
<a href="#">Edit Columns</a>	
Theme	<input type="text" value="aristo"/>
<input type="checkbox"/> Use CDN	
CDN path	<input type="text" value="http://cdn.wijmo.com/"/>
<a href="#">About...</a>	