
ComponentOne

ProgressBar for ASP.NET Wijmo

Copyright © 2012 ComponentOne LLC. All rights reserved.

Corporate Headquarters

ComponentOne LLC

201 South Highland Avenue
3rd Floor
Pittsburgh, PA 15206 • USA

Internet: info@ComponentOne.com

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

Sales

E-mail: 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 ProgressBar 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 ProgressBar for ASP.NET Wijmo Components to a Project | 14 |
| Key Features | 15 |
| Wijmo Top Tips | 15 |
| ProgressBar for ASP.NET Wijmo Quick Start | 16 |
| Step 1 of 3: Creating the Project and Adding Controls | 16 |
| Step 2 of 3: Configuring the Controls | 16 |
| Step 3 of 3: Running the Project | 17 |
| C1ProgressBar Design-Time Support | 18 |
| C1ProgressBar Smart Tag and Tasks Menu | 18 |
| C1ProgressBar Context Menu | 19 |
| C1ProgressBar Elements | 19 |
| Track | 20 |

| | |
|---|----|
| Progress Indicator | 20 |
| Label | 20 |
| C1ProgressBar Appearance and Behavior | 21 |
| Themes | 21 |
| Keyboard Access..... | 22 |
| ToolTips | 22 |
| Easing Effect Descriptions | 22 |
| Animation Duration | 24 |
| ProgressBar for ASP.NET Wijmo Task-Based Help | 24 |
| Customizing the C1ProgressBar Label | 24 |
| Aligning the C1ProgressBar Label | 24 |
| Formatting the C1ProgressBar Label..... | 25 |
| Working with Themes | 26 |
| Using a Built-In Theme..... | 27 |
| Using a Custom Theme | 27 |
| Formatting the C1ProgressBar ToolTip..... | 29 |
| Changing the Progress Indicator Fill Direction..... | 30 |
| Configuring C1ProgressBar Animations | 31 |
| Setting C1ProgressBar Values..... | 33 |
| ProgressBar for ASP.NET Wijmo Client-Side Reference | 34 |
| Using the Wijmo CDN | 35 |

ComponentOne ProgressBar for ASP.NET Wijmo Overview

Keep your end-users informed by adding **ComponentOne ProgressBar for ASP.NET Wijmo** to your next Web project. **ProgressBar for ASP.NET Wijmo** is a graphical user-interface element that serves as both a gauge and an indicator; it allows user determine the progress of an operation while also reminding them that the process is still running. You can use it to present a static symbol of progress or to illustrate the progress of a live operation. **ProgressBar for ASP.NET Wijmo** features five built-in visual styles, customizable labels and ToolTips, dozens of animations, and much more.

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**

- [Quick Start](#) (page 16)
- [Design-Time Support](#) (page 18)

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:

| | |
|--------------|---|
| Bin | Contains copies of all binaries (DLLs, Exes) in the ComponentOne Visual Studio ASP.NET Wijmo package. |
| Wijmo | 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 Viewer help to the C:\Program Files\ComponentOne\Studio for ASP.NET Wijmo directory in the following folder:

| | |
|-------------------|---|
| HelpViewer | 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:

| | |
|---------------------------------|---|
| Common | Contains support and data files that are used by many of the demo programs. |
| Studio for ASP.NET Wijmo | 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:

| | |
|---------------------------|---|
| Operating Systems: | Windows Server® 2003 Windows Server 2008 Windows XP SP2 Windows Vista™ Windows 7 |
| Web Server: | Microsoft Internet Information Services (IIS) 6.0 or later |
| Environments: | .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)**.
2. Select **ComponentOne Studio for ASP.NET Wijmo Help** and click the **Remove** button.
3. 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>
</NamedPermissionSets>
```

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 either doesn't exist, contains wrong information, or is not 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.
8. Paste the line from step 5 into this file (replace the old licensing information with the new).
9. 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 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

3rd 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 **C1ProgressBar** using the fully qualified name for this class:

- Visual Basic

```
Dim MaskedInput As C1.Web.Wijmo.Controls.C1ProgressBar.C1ProgressBar
```

- C#

```
C1.Web.Wijmo.Controls.C1Input.C1ProgressBar. C1ProgressBar;
```

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 C1ProgressBar = C1.Web.UI.Controls.C1ProgressBar.C1ProgressBar  
Imports MyProgressBar = MyProject.Objects.C1ProgressBar.C1ProgressBar
```

```
Dim wm1 As C1ProgressBar
```

```
Dim wm2 As MyProgressBar
```

- C#

```
using C1ProgressBar = C1.Web.UI.Controls.C1ProgressBar.C1ProgressBar;  
using MyProgressBar = MyProject.Objects.C1ProgressBar.C1ProgressBar;
```

```
C1ProgressBar wm1;
```

```
MyProgressBar 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 CodePlex 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 http://www.asp.net/ajax/downloads/archive/ . 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 ProgressBar for ASP.NET Wijmo Components 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 **C1ProgressBar** component will appear in the Visual Studio Toolbox customization dialog box.

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.
2. 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.
3. 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.C1ProgressBar. Note that there may be more than one component for each namespace.
4. 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 C1.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

The following are some of the main features of C1ProgressBar that you may find useful:

- **Animations**
Choose from over 30 built-in animations to spice up the progress bar's fill effect. You can choose how fast and how frequently each animation runs.
- **Theming**
With just a click of the SmartTag, change the progress bar'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.
- **Customizable Labels**
Customize your label so that it appears on right/bottom, left/top, or in the center of the control. Labels can also be set to run as a marquee as the progress bar fills. Labels can be formatted using one of six progress indicators: current progress value, current percent progress, remaining progress, percentage remaining, minimum value, and maximum value. See [Label](#) (page 20) for more information.
- **CSS Support**
Use a cascading style sheet (CSS) style to define custom skins. CSS support allows you to match the progressbar to your organization's standards.

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" />
```

ProgressBar for ASP.NET Wijmo

Quick Start

In this **ProgressBar for ASP.NET Wijmo** quick start, you will create an ASP.NET Web site project with one C1ProgressBar and two **Button** controls. When the **Start** button is clicked, it fires the RunTask event, and the progress bar will start updating, via the UpdateProgress method, every half second (500 milliseconds). When the **Stop** button is clicked, the progress bar will no longer update.

Step 1 of 3: Creating the Project and Adding Controls

In this step of the quick start, you will create your ASP.NET Web site project and add a C1ProgressBar and two **Button** controls to the page.

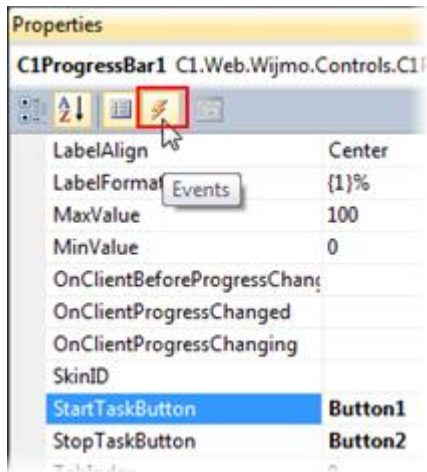
1. Create a new [ASP.NET Web site](#) (page 13).
2. Click the **Design** button to enter Design view.
3. Double-click the C1ProgressBar icon in the Visual Studio Toolbox to add the control to your page.
4. Add two standard **Button** controls, setting the **Text** properties to **Start** and **Stop** respectively, so the page looks similar to the following example:



Step 2 of 3: Configuring the Controls

In this step, you will specify the StartTaskButton and StopTaskButton, which will designate the button controls as the buttons that will start and stop the progress bar updating process.

1. Select the C1ProgressBar control and in the Visual Studio Properties window, click the drop-down arrow next to StartTaskButton and select **Button1**.
2. Set the StopTaskButton property to **Button2**.
3. In the Properties window, click the **Events** button to list all of the events for the **C1ProgressBar1** control.



Next to the RunTask event, enter **ProgressBar1_RunTask**. This creates an event in Code view, where you can enter the following code, so the event looks like this:

- Visual Basic

```
Protected Sub ProgressBar1_RunTask(sender As Object, e As
C1.Web.Wijmo.Controls.C1ProgressBar.C1ProgressBarTaskEventArgs)
    For i As Integer = 0 To 99
        System.Threading.Thread.Sleep(500)
        e.UpdateProgress(i)
    Next
End Sub
```

- C#

```
protected void ProgressBar1_RunTask(object sender,
C1.Web.Wijmo.Controls.C1ProgressBar.C1ProgressBarTaskEventArgs e)
{
    for (int i = 0; i < 100; i++)
    {
        System.Threading.Thread.Sleep(500);
        e.UpdateProgress(i);
    }
}
```

When the **Start** button is clicked, the RunTask event fires and calls the UpdateProgress method to update the progress bar by 1 integer every 500 milliseconds.

Step 3 of 3: Running the Project

Press **F5** to run the project and click the Start button. The C1ProgressBar starts updating 1% every half of a second. Click the **Stop** button to stop the updates.



C1ProgressBar Design-Time Support

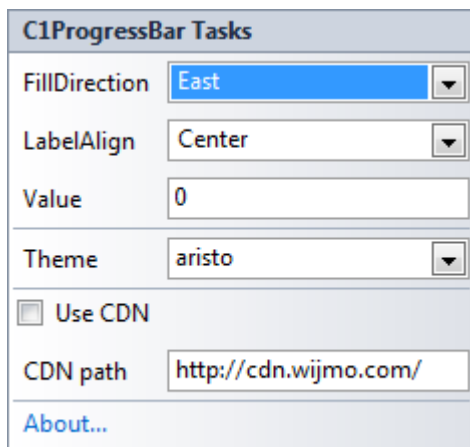
C1ProgressBar provides smart tags and a designer that offers rich design-time support and simplifies working with the object model.

The following topics describe how to use **C1ProgressBar**'s design-time environment to configure the **C1ProgressBar** control.

C1ProgressBar Smart Tag and Tasks Menu

The **C1ProgressBar** control includes a smart tag in Visual Studio. A smart tag represents a short-cut tasks menu that provides the most commonly used properties in **C1ProgressBar**.

To access the **C1ProgressBar Tasks** menu, click on the smart tag in the upper-right corner of the **C1ProgressBar** control. This will open the **C1ProgressBar Tasks** menu.



The **C1ProgressBar Tasks** menu operates as follows:

- **FillDirection**
The direction from which the progress indicator fills the progress bar.
- **LabelAlign**
The alignment of the progress bar's progress label.
- **Value**
The value of the progress indicator. This value must be somewhere between the **MinValue** and **MaxValue** property.
- **Theme**
The Wijmo theme for the control. You can select from one of the five built-in controls, or you may enter the location of your custom theme into this box.
- **Use CDN**
The **Use CDN** check box specifies whether you want to get the client-side reference from an online CDN.
- **CDN Path**
The path to the CDN files.

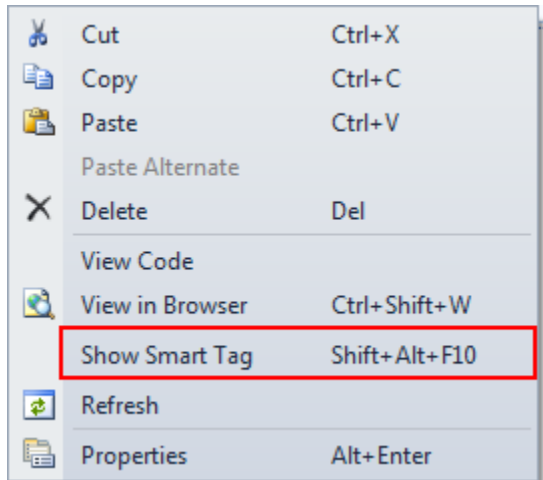
- **About**

The **About** link brings up a screen that provides information about the company and the product.

C1ProgressBar Context Menu

C1ProgressBar has additional commands available on the context menu that Visual Studio provides for all .NET and ASP.NET controls.

Right-click anywhere on the C1ProgressBar control to display the context menu:



The **C1ProgressBar** context menu operates as follows:

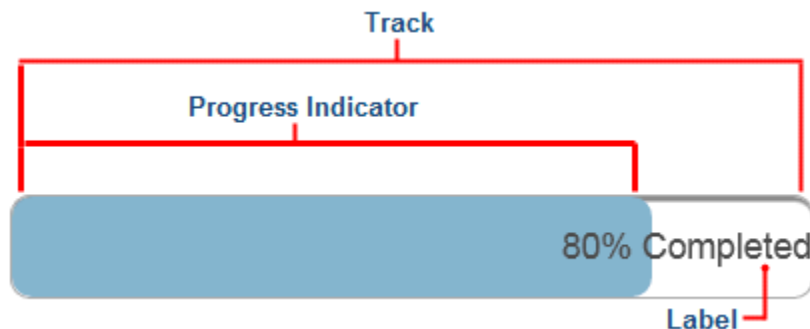
- **Show Smart Tag**

Clicking **Show Smart Tag** opens the **C1ProgressBar Tasks** menu.

C1ProgressBar Elements

The C1ProgressBar control is a graphical user-interface element that provides users with a visual representation of the progress of an operation. It is different from most of the other control in Studio for ASP.NET AJAX in that it isn't meant to be interactive; it merely indicates that a process is running.

C1ProgressBar, by default, consists of three different elements: the track, the progress indicator, and a label. These elements are labeled in the following graphic:



- **Track:** The track runs the length of the control and contains the label control. When a process is started, the track will also become home to the progress indicator.
- **Progress Indicator:** The progress indicator provides a visual representation of the progress completed in a task.
- **Label:** The label control is used to provide an alpha-numeric representation of progress. The label can show six different values; it can also be overridden to display static text or static numbers.

The following topics illustrate the elements of the C1ProgressBar.

Track

The C1ProgressBar control's track runs the length of the control. By default, the track appears in only one color (usually gray), indicating that no process has been started yet. If a process has been started, track will appear in two contrasting colors, one of which indicates the amount of the process that has completed and the other of which indicates the amount of the process that is left.

The size of the C1ProgressBar track is set using the Width and Height properties. The track can be vertical or horizontal in orientation.

Progress Indicator

The progress indicator provides a visual representation of the current progress between a minimum and a maximum value. The progress indicator can fill from the right, left, top or side by setting the FillDirection property. The FillDirection property can be set as follows:

| Setting | Description |
|---------|---|
| North | The progress indicator fills from the bottom to the top of the control This switches the orientation of the control to a vertical position. |
| South | The progress indicator fills from the top to the bottom of the control This switches the orientation of the control to a vertical position. |
| East | The progress indicator fills from right to left of the control |
| West | The progress indicator fills from left to right of the control |

The progress indicator is filled to a percentage, which is based the value of the Value property as compared to the values of the MinValue and MaxValue properties. For example, if the MinValue property is set to 0, the MaxValue is set to 100, and the Value property is set to 25, the progress indicator will fill 25% of the track. If the MinValue property is set to 100, the MaxValue is set to 300, and the Value property is set to 200, the progress indicator will fill 50% of the track.

The progress indicator can represent either a fixed indication of progress, or it can be used to convey up-to-date progress indicators.

Label

The progress bar's label is used to provide an alpha-numeric representation of progress. The label can display any one of these six formats:

| Topic | Description |
|--------------------------|--|
| {0} or {ProgressValue} | Displays the current progress. |
| {1} or {PercentProgress} | Displays the current percent progress. |

| | |
|------------------------------|--|
| {2} or {RemainingProgress} | Displays the progress needed to complete the task. |
| {3} or {PercentageRemaining} | Displays the percentage of the process remaining. |
| {4} or {Min} | Displays the value of the MinValue property. |
| {5} or {Max} | Displays the value of the MaxValue property. |

To set the label to one of these formats, set the LabelFormatString property. You can add any text you want around outside of the curly braces. For example, you can enter “{ProgressValue} percent finished” and that will show the percentage that is left to complete in the task. You can also overwrite the entire ToolTip with a custom string.

Label Alignment

The label element can be aligned in several different ways by setting the LabelAlign property. The following settings are available:

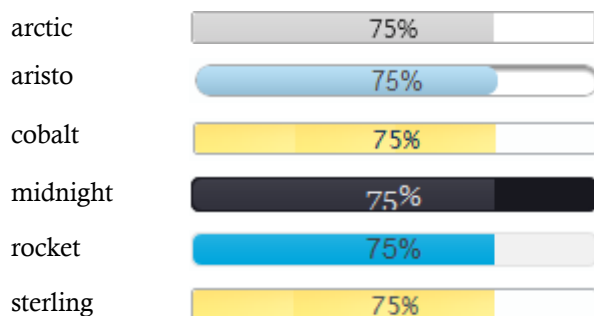
| Setting | Description |
|---------|--|
| East | The label is aligned to the right side of the control. |
| West | The label is aligned to the left side of the control. |
| Center | The label is aligned to the center of the control. |
| North | The label is aligned to the top of the control. |
| South | The label is aligned to the bottom of the control. |
| Running | The label runs as a marquee. |

C1ProgressBar Appearance and Behavior

The following topics provide information about C1ProgressBar's behavioral and appearance-related features. Some of these features affect how the control acts when loaded, whereas others affect the users' interactions with the control.

Themes

The C1ProgressBar control contains six built-in themes. When one of these themes is selected, all other ASP.NET Wijmo studio controls on the page will be skinned accordingly. The themes will appear on the C1ProgressBar control as follows:



To set the theme of the C1ProgressBar control, simply set its Theme property to one of the built-in themes.

Keyboard Access

ProgressBar for ASP.NET AJAX features keyboard support for the C1ProgressBar control. You can enable this feature for the whole control by setting the **C1ProgressBar.AccessKey** property to an access key. Once the **AccessKey** property is set, you can access the control by pressing the ALT key and the access key simultaneously on your keyboard.

ToolTips

You can use the ToolTip property to create a user-friendly interface. ToolTips are graphic user interface elements that are used to provide users with information regarding a UI element. When users hover over the element with their cursor, a box will appear with the additional information.

The C1ProgressBar control's ToolTip is usually used to inform users about the process that is taking place; it lets users know how much longer the process will be.

The C1ProgressBar control's ToolTip can display any one of these six formats:

| Topic | Description |
|------------------------------|--|
| {0} or {ProgressValue} | Displays the current progress. |
| {1} or {PercentProgress} | Displays the current percent progress. |
| {2} or {RemainingProgress} | Displays the progress needed to complete the task. |
| {3} or {PercentageRemaining} | Displays the percentage of the process remaining. |
| {4} or {Min} | Displays the value of the MinValue property. |
| {5} or {Max} | Displays the value of the MaxValue property. |

Easing Effect Descriptions

C1ProgressBar contains over thirty built-in animation effects that change the reaction of the progress bar when it is moved. The default easing effect is **Swing**, but you can set it to another effect using the Easing property. You can also turn all animations off by setting the **Animation.Disabled** property to **True**.

The table below describes each animation effect:

| Name | Description |
|-----------------|--|
| Linear | Linear easing. Moves smoothly without acceleration or deceleration. |
| Swing (default) | This is the default animation effect. |
| EaseInQuad | Quadratic easing in. Starts slowly and then accelerates. |
| EaseOutQuad | Quadratic easing out. Starts quickly and then decelerates. |
| EaseInOutQuad | Quadratic easing in and out. Starts slowly, accelerates, and then decelerates. |
| EaseInCubic | Cubic easing in. Starts slowly and then accelerates. |
| EaseOutCubic | Cubic easing out. Starts quickly and then decelerates. |

| | |
|------------------|--|
| EaseInOutCubic | Cubic easing in and out. Starts slowly, accelerates, and then decelerates. |
| EaseInQuart | Quartic easing in. Starts slowly and then accelerates. |
| EaseOutQuart | Quartic easing out. Starts quickly and then decelerates. |
| EaseInOutQuart | Quartic easing in and out. Starts slowly, accelerates, and then decelerates. |
| EaseInQuint | Quintic easing in. Starts slowly and then accelerates. |
| EaseOutQuint | Quintic easing out. Starts quickly and then decelerates. |
| EaseInOutQuint | Quintic easing in and out. Starts slowly, accelerates, and then decelerates. |
| EaseInSine | Sinusoidal easing in. Starts slowly and then accelerates. |
| EaseOutSine | Sinusoidal easing out. Starts quickly and then decelerates. |
| EaseInOutSine | Sinusoidal easing in and out. Starts slowly, accelerates, and then decelerates. |
| EaseInExpo | Exponential easing in. Starts slowly and then accelerates. |
| EaseOutExpo | Exponential easing out. Starts quickly and then decelerates. |
| EaseInOutExpo | Exponential easing in and out. Starts slowly, accelerates, and then decelerates. |
| EaseInCirc | Circular easing in. Starts slowly and then accelerates. |
| EaseOutCirc | Circular easing out. Starts quickly and then decelerates. |
| EaseInOutCirc | Circular easing in and out. Starts slowly, accelerates, and then decelerates. |
| EaseInElastic | Elastic easing in. Starts slowly and then accelerates. |
| EaseOutElastic | Elastic easing out. Starts quickly and then decelerates. |
| EaseInOutElastic | Elastic easing in and out. Starts slowly, accelerates, and then decelerates. |
| EaseInBack | Back easing in. Starts slowly and then accelerates. |
| EaseOutBack | Back easing out. Starts quickly and then decelerates. |
| EaseInOutBack | Back easing in and out. Starts slowly, accelerates, and then decelerates. |
| EaseInBounce | Bouncing easing in. Starts slowly and then accelerates. |
| EaseOutBounce | Bouncing easing out. Starts quickly and then decelerates. |
| EaseInOutBounce | Bouncing easing in and out. Starts slowly, accelerates, and then decelerates. |

Animation Duration

You can set the length of **C1Splitter**'s animation effect takes using the `Duration` property. The unit of time used for specifying animation effect duration is in milliseconds, and the default setting for the `Duration` property is **500** milliseconds (or half a second). Increase this value for longer animation effect, and decrease this number for a shorter animation effect.

ProgressBar for ASP.NET Wijmo Task-Based Help

The task-based help section assumes that you are familiar with programming in the Visual Studio ASP.NET environment and have a general understanding of the **ComponentOne ProgressBar** control. Each topic provides a solution for specific tasks using the `C1ProgressBar` control. By following the steps outlined in each topic, you will be able to create projects using a variety of `C1ProgressBar` features. Each task-based help topic also assumes that you have created a new ASP.NET project.

Customizing the C1ProgressBar Label

The following topics discuss how to customize the `C1ProgressBar` control's label element.

Aligning the C1ProgressBar Label

By default, the `C1ProgressBar` control's label is aligned in the center of the control. In this topic, you will learn how to change the alignment of the label in Design view, in Source view, and in code.

Aligning the Label in Design View

Complete the following steps:

1. Click the **Design** button to enter Design view.
2. Right-click the `C1ProgressBar` control to open its context menu and select **Properties**.
The Properties window opens with the `C1ProgressBar` control's properties in focus.
3. Locate the `LabelAlign` property, click its drop-down arrow, and select **East** from the list.
4. Press F5 to run the project and observe that the label now appears on the right side of the control.

Aligning the Label in Source View

Complete the following steps:

1. Click the **Source** button to enter Source view.
2. Add `LabelAlign = "East"` to the `<wijmo:C1ProgressBar>` tag so that the markup resembles the following:

```
<wijmo:C1ProgressBar ID="C1ProgressBar1" runat="server" LabelAlign="East" />
```

3. Press F5 to run the project and observe that the label now appears on the right side of the control.

Aligning the Label in Code

Complete the following steps:

1. On the Visual Studio toolbar, click **View | Code** to enter code view.
2. Import the following namespace into your project:

- Visual Studio

```
Imports C1.Web.Wijmo.Controls.C1ProgressBar
```

- C#

```
using C1.Web.Wijmo.Controls.C1ProgressBar;
```

3. Align the label by placing the following code in the **Page_Load** event:

- Visual Studio

```
C1ProgressBar1.LabelAlign = LabelAlign.East
```

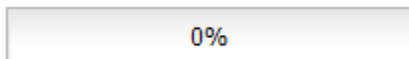
- C#

```
C1ProgressBar1.LabelAlign = LabelAlign.East;
```

4. Press **F5** to run the project and observe that the label now appears on the right side of the control.

✔ This Topic Illustrates the Following:

In this topic, you aligned the C1ProgressBar control's label to the right side of the control. The result of this topic resembles the following:



Formatting the C1ProgressBar Label

By default, the C1ProgressBar control label displays a label with the current percentage of progress. In this topic, you will customize the label so that it displays a modified string along with the maximum value setting of the control. This topic assumes that you have created an ASP.NET AJAX-Enabled Web site containing a **ScriptManager** control and a C1ProgressBar control.

Formatting the Label in Design View

Complete the following steps:

1. Click the **Design** button to enter Design view.
2. Right-click the C1ProgressBar control to open its context menu and select **Properties** from the list.
The Properties window opens with the C1ProgressBar control's properties in focus.
3. Set the LabelFormatString property to "Maximum Value: {5}".



4. Press **F5** to run the project and run your cursor over the `C1ProgressBar` control. Observe that a label appears with “Maximum value: 100” as its string.

Formatting the Label in Source View

Complete the following steps:

1. Click the **Source** button to enter Source view.
2. Place `LabelFormatString = "Maximum Value: {5}"` in the `<wijmo:C1ProgressBar>` tag so that the markup resembles the following:

```
<wijmo:C1ProgressBar ID="C1ProgressBar1" runat="server"
Width="288px" Height="22px" UseEmbeddedVisualStyles="True"
VisualStyle="ArcticFox" LabelFormatString="Maximum value: {5}" />
```

3. Press **F5** to run the project and run your cursor over the `C1ProgressBar` control. Observe that a label appears with “Maximum value: 100” as its string.

Formatting the Label in Code

Complete the following steps:

1. On the Visual Studio toolbar, click **View | Code** to enter code view.
2. Import the following namespace into your project:

- Visual Studio
`Imports C1.Web.Wijmo.Controls.C1ProgressBar`

- C#
`using C1.Web.Wijmo.Controls.C1ProgressBar;`

3. Customize the label by adding the following code to the **Page_Load** event:

- Visual Studio
`C1ProgressBar1.LabelFormatString = "Maximum value: {5}"`

- C#
`C1ProgressBar1.LabelFormatString = "Maximum value: {5}";`

4. Press **F5** to run the project and run your cursor over the `C1ProgressBar` control. Observe that a label appears with “Maximum value: 100” as its string.

✔ This Topic Illustrates the Following:

In this topic, you created a custom label for the `C1ProgressBar` control. The following topic illustrates a progress bar with its `LabelFormatString` property set to “Maximum Value: {5}”.



Working with Themes

The topics in this section illustrate how to utilize built-in themes and custom themes.

Using a Built-In Theme

A `C1ProgressBar` control has six embedded themes that you can apply with just a few clicks. This topic illustrates how to change the theme in Design view, in Source view, and in code. For more information on themes, see [Themes](#) (page 21).

Changing the Theme in Design View

Complete the following steps:

1. Click the **C1ProgressBar** smart tag (📌) to open the **C1ProgressBar Tasks** menu.
2. Click the **Theme** drop-down arrow and select a theme from the list. For this example, select **rocket**.

The **rocket** theme is applied to the `C1ProgressBar` control.

Changing the Theme in Source View

To change the theme of your `C1ProgressBar` in Source view, add `VisualStyle="rocket"` to the `<wijmo:C1ProgressBar>` tag so that it resembles the following:

```
<wijmo:C1ProgressBar ID="C1ProgressBar1" runat="server" Theme="rocket"/>
```

Changing the Theme in Code

Complete the following steps:

1. Import the following namespace into your project:

- Visual Basic
`Imports C1.Web.Wijmo.Controls`

- C#
`using C1.Web.Wijmo.Controls;`

2. Add the following code, which sets the Theme property, to the **Page_Load** event:

- Visual Basic
`C1ProgressBar1.Theme = "rocket"`

- C#
`C1ProgressBar1.Theme = "rocket";`

3. Run the program.

✔ **This topic illustrates the following:**

The following image shows a `C1ProgressBar` control with the **rocket** theme:



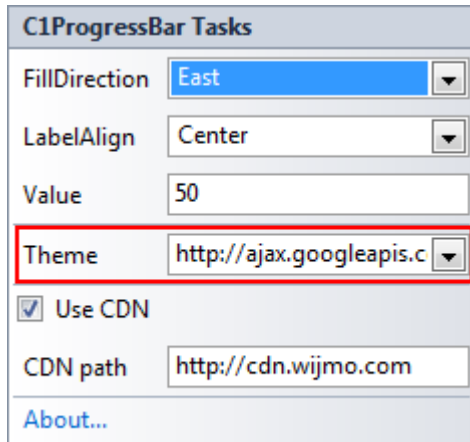
Using a Custom Theme

`ProgressBar` 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. We will use `C1ProgressBar` in the following examples.

Using a CDN URL

Complete the following steps:

1. Click the C1ProgressBar smart tag to open the **Tasks** menu.
2. 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 *trontastic* theme: <http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.14/themes/trontastic/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.

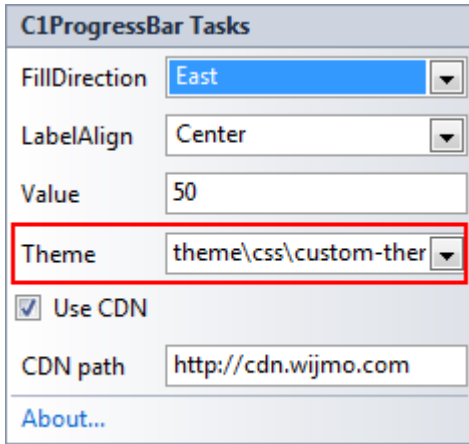
3. Run the project and notice the theme is applied to C1ProgressBar.



Using jQuery ThemeRoller

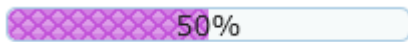
Complete the following steps:

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 **customtheme** folder.
5. In the Solution Explorer, click **Show All Files** and then right-click the **customtheme** folder and select **Include in Project**.
6. Click the C1ProgressBar smart tag to open the **Tasks** menu.
7. In the **Theme** property, enter the path to your custom theme .css; for example, **custom-theme\css\custom-theme/jquery-ui-1.8.15.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.

- Run the project and notice the theme is applied to C1ProgressBar.



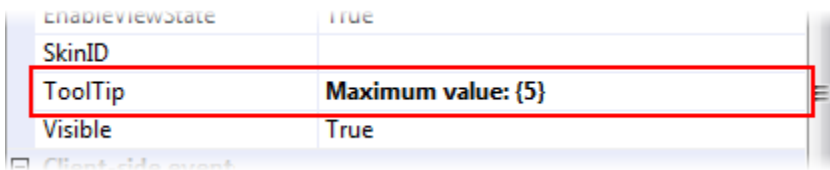
Formatting the C1ProgressBar ToolTip

At run time, the C1ProgressBar control will display a ToolTip if a user runs his cursor over it. By default, this ToolTip will display the current percentage of progress. In this topic, you will customize the ToolTip so that it displays a modified string along with the maximum value setting of the control. This topic assumes that you have created an ASP.NET AJAX-Enabled Web site containing a `ScriptManager` control and a C1ProgressBar control.

Formatting the ToolTip in Design View

Complete the following steps:

- Click the **Design** button to enter Design view.
- Right-click the C1ProgressBar control to open its context menu and select **Properties** from the list.
The Properties window opens with the C1ProgressBar control's properties in focus.
- Set the ToolTip property to "Maximum Value: {5}".



- Press **F5** to run the project and run your cursor over the C1ProgressBar control. Observe that a ToolTip appears with "Maximum value: 100" as its string.

Formatting the ToolTip in Source View

Complete the following steps:

- Click the **Source** button to enter Source view.

- Place `ToolTip = "Maximum Value: {5}"` in the `<wijmo:C1ProgressBar>` tag so that the markup resembles the following:

```
<wijmo:C1ProgressBar ID="C1ProgressBar1" runat="server"
Width="288px" Height="22px" UseEmbeddedVisualStyles="True"
VisualStyle="ArcticFox" ToolTip="Maximum value: {5}" />
```

- Press F5 to run the project and run your cursor over the C1ProgressBar control. Observe that a ToolTip appears with "Maximum value: 100" as its string.

Formatting the ToolTip in Code

Complete the following steps:

- On the Visual Studio toolbar, click **View | Code** to enter code view.
- Import the following namespace into your project:
 - Visual Studio
`Imports C1.Web.Wijmo.Controls.C1ProgressBar`
 - C#
`using C1.Web.Wijmo.Controls.C1ProgressBar;`
- Customize the ToolTip by adding the following code to the **Page_Load** event:

- Visual Studio
`C1ProgressBar1.ToolTip = "Maximum value: {5}"`

- C#
`C1ProgressBar1.ToolTip = "Maximum value: {5}";`

- Press F5 to run the project and run your cursor over the C1ProgressBar control. Observe that a ToolTip appears with "Maximum value: 100" as its string.

✔ This Topic Illustrates the Following:

In this topic, you created a custom ToolTip for the C1ProgressBar control. The following image illustrates a progress bar with its ToolTip property set to "Maximum Value: {5}".



Changing the Progress Indicator Fill Direction

By default, the progress indicator will fill the track starting from the left side of the control. The fill direction can be changed to fill from the right, bottom, or top by setting one property: **FillDirection**. In this topic, you will set the **FillDirection** property in Design view, in Source view, and in code.

Changing the Fill Direction in Design View

Complete the following steps:

1. Click the **Design** button to enter Design view.
2. Right-click the C1ProgressBar control to open its context menu and select **Properties**.
The Properties window opens with the C1ProgressBar control's properties in focus.
3. In the Properties window, set the **FillDirection** property. For this example, set the property to **West**.

Changing the Fill Direction in Source View

Complete the following steps:

1. Click the **Source** button to enter Source view.
2. Add `FillDirection="West"` to **C1ProgressBar**'s tag so that the markup resembles the following:

```
<wijmo:C1ProgressBar ID="C1ProgressBar1" runat="server"  
FillDirection = "West" Value="100" AnimationDuration="6000"/>
```

Changing the Fill Direction in Code

Complete the following steps:

1. On the Visual Studio toolbar, click **View | Code** to enter code view.
2. Import the following namespace into your project:

- Visual Studio
`Imports C1.Web.Wijmo.Controls.C1ProgressBar`

- C#
`using C1.Web.Wijmo.Controls.C1ProgressBar;`

3. Add the following code to the **Page_Load** event:

- Visual Basic
`'Set the FillDirection property to FromRightOrBottom
C1ProgressBar1.FillDirection = FillDirection.West`

- C#
`// Set the FillDirection property to FromRightOrBottom
C1ProgressBar1.FillDirection = FillDirection.West;`

Configuring C1ProgressBar Animations

The C1ProgressBar control features thirty-one built in animations. In this topic, you will set the animation, the duration of the animation, and the delay of the animation in Design view, in Source view, and in code.

Configuring Animations in Design View

Complete the following steps:

1. Click the **Design** button to enter Design view.
2. Right-click the C1ProgressBar control to open its context menu and select **Properties**.

The Properties window opens with the C1ProgressBar control's properties in focus.

3. In the Properties window, set the following properties:
 - Set the **AnimationOptions.Easing** property to **EaseOutBounce**. This establishes the animation effect.
 - Set the **AnimationOptions.Duration** property to "6000". This establishes the length of the animation.
 - Set the **AnimationDelay** property to "500". This establishes the length of time that will pass before the animation starts.
 - Set the **Value** property to "100". This will cause the progress indicator to fill the track at run time.
4. Press **F5** to run the project and observe that the animation takes about a half-second to begin and six seconds to complete. The animation, **EaseOutBounce**, causes the progress indicator to bounce against the end of the track before settling into its resting state.

Configuring Animations in Source View

Complete the following steps:

1. Click the **Source** button to enter Source view.
2. Add `AnimationDelay="500"` to the `<wijmo:C1ProgressBar>` tag so that the markup resembles the following:

```
<wijmo:C1ProgressBar ID="C1ProgressBar1" runat="server"
AnimationDelay="500"></wijmo:C1ProgressBar>
```

3. Place the following markup between the `<wijmo:C1ProgressBar>` tags:

```
<AnimationOptions Duration="6000" Easing="EaseOutBounce" />
```

4. Press **F5** to run the project and observe that the animation takes about a half-second to begin and six seconds to complete. The animation, **EaseOutBounce**, causes the progress indicator to bounce against the end of the track before settling into its resting state.

Configuring Animations in Code

Complete the following steps:

1. On the Visual Studio toolbar, click **View | Code** to enter code view.
2. Add the following code to the **Page_Load** event:

- Visual Basic

```
`Set animation properties
C1ProgressBar1.AnimationDelay = 500;
C1ProgressBar1.AnimationOptions.Duration = 6000;
C1ProgressBar1.AnimationOptions.Easing =
C1.Web.Wijmo.Controls.Easing.EaseOutBounce;
`Set Value property so that the progress indicator automatically fills
at run time
C1ProgressBar1.Value = 100
```

- C#

```
//Set animation properties
C1ProgressBar1.AnimationDelay = 500;
```

```

C1ProgressBar1.AnimationOptions.Duration = 6000;
C1ProgressBar1.AnimationOptions.Easing =
C1.Web.Wijmo.Controls.Easing.EaseOutBounce;

//Set Value property so that the progress indicator automatically fills at run
time
C1ProgressBar1.Value = 100;

```

3. Press **F5** to run the project and observe that the animation takes about a half-second to begin and six seconds to complete. The animation, **EaseOutBounce**, causes the progress indicator to bounce against the end of the track before settling into its resting state.

Setting C1ProgressBar Values

The C1ProgressBar control contains a minimum value, a maximum value, and a value. In this topic, you will change the value of the MinValue, MaxValue, and Value properties from their defaults (0, 100, and 0 respectively) in Design view, in Source view, and in code.

Setting Values in Design View

Complete the following steps:

1. Click the **Design** button to enter Design view.
2. Right-click the C1ProgressBar control to open its context menu and select **Properties**.
The Properties window opens with the C1ProgressBar control's properties in focus.
3. In the Properties window, set the following properties:
 - Set the MaxValue property to "300".
 - Set the MinValue property to "100".
 - Set the Value property to "150".
4. Press **F5** to run the project and observe that the progress indicator has progressed along a quarter of the task bar. It has increased by a quarter because the control's value (150) is one-quarter of the way between the minimum value (100) and the maximum value (300).

Setting Values in Source view

Complete the following steps:

1. Click the **Source** button to enter Source view.
2. Add `MinimumValue="100"`, `MaximumValue="300"`, and `Value="150"` to the `<wijmo:C1ProgressBar>` tag so that the markup resembles the following:

```

<wijmo:C1ProgressBar ID="C1ProgressBar1" runat="server"
ToolTip="Maximum value: {5}" LabelFormatString="{0}" MaximumValue="300"
MinimumValue="100" Value="150" />

```

3. Press **F5** to run the project and observe that the progress indicator has progressed along a quarter of the task bar. It has increased by a quarter because the control's value (150) is one-quarter of the way between the minimum value (100) and the maximum value (300).

Setting Values in Code

Complete the following steps:

1. To configure animations, complete the following steps:
2. On the Visual Studio toolbar, click **View | Code** to enter code view.
3. To set the MaxValue property, add the following code to the **Page_Load** event:
 - Visual Basic

```
C1ProgressBar1.MaxValue = 300
```
 - C#

```
C1ProgressBar1.MaxValue = 300;
```
4. To set the MinValue property, add the following code to the **Page_Load** event:
 - Visual Basic

```
C1ProgressBar1.MinValue = 100
```
 - C#

```
C1ProgressBar1.MinValue = 100;
```
5. To set the Value property, add the following code to the **Page_Load** event:
 - Visual Basic

```
C1ProgressBar1.Value = 150
```
 - C#

```
C1ProgressBar1.Value = 150;
```
6. Press **F5** to run the project and observe that the progress indicator has progressed along a quarter of the task bar. It has increased by a quarter because the control's value (150) is one-quarter of the way between the minimum value (100) and the maximum value (300).

ProgressBar 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 **ProgressBar 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.

- [ProgressBar Options](#)
- [ProgressBar Events](#)

- [ProgressBar Methods](#)

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.

CSAccordion Tasks

Choose Data Source: (None) ▾

ExpandDirection: Bottom ▾

Edit Pages

[Add new page](#)

[Remove selected page](#)

SelectedIndex: 0

Theme: aristo ▾

Use CDN

CDN path:

[About...](#)