ComponentOne

ProgressBar for ASP.NET AJAX

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 AJAX Overview

Keep your end-users informed by adding **ComponentOne ProgressBar for ASP.NET AJAX** to your next Web project. **ProgressBar for ASP.NET AJAX** 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 AJAX** features five built-in visual styles, customizable labels and ToolTips, dozens of animations, and much more.

ProgressBar for ASP.NET AJAX is part of **ComponentOne Studio for ASP.NET AJAX**, the next breed of ASP.NET controls developed on a new client and server-side framework. This new ASP.NET control suite fully exploits the AJAX framework to enable you to create highly interactive and sophisticated Web applications with Studio for ASP.NET AJAX.

Getting Started

- Quick Start (page 16)
- Design-Time Support (page 22)
- Task-Based Help (page 30)

What's New in ProgressBar for ASP.NET AJAX

No new features have been added to the **C1ProgressBar** control in its 2010 v2 release.

Tip: A version history containing a list of new features, improvements, fixes, and changes for each product is available at HelpCentral at http://helpcentral.componentone.com/VersionHistory.aspx.

Installing ProgressBar for ASP.NET AJAX

The following sections provide helpful information on installing **ComponentOne ProgressBar for ASP.NET AJAX**:

ProgressBar for ASP.NET AJAX Setup Files

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

bin	Contains copies of all binaries (DLLs, Exes) in the ComponentOne Visual Studio ASP.NET package.
H2Help	Contains documentation for Studio for ASP.NET AJAX components.
C1WebUi	Contains files (at least a readme.txt) related to the product.
C1WebUi\VisualStyles	Contains all external file themes.

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\<username>\My Documents\ComponentOne Samples

Windows 7/Vista path: C:\Users\<username>\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\C1WebUi	Contains a readme.txt file and the folders that make up the Control Explorer and other samples.

You can access samples from the **ComponentOne Control Explorer**. To view samples, click the **Start** button and then click **ComponentOne | Studio for ASP.NET | Control Explorer**.

System Requirements

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

Operating Systems:	Windows® 2000
	Windows Server® 2003
	Windows Server 2008
	Windows XP SP2
	Windows Vista™
	Windows 7
Web Server:	Microsoft Internet Information Services (IIS) 5.0 or later
Environments:	.NET Framework 2.0 or later
	Visual Studio 2005 or Visual Studio 2008
	Internet Explorer 6.0 or later
	Firefox® 2.0 or later
	Safari® 2.0 or later
Disc Drive:	CD or DVD-ROM drive if installing from CD

Uninstalling ProgressBar for ASP.NET AJAX

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

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

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 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 exiting 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 exiting web_mediumtrust.config file. To edit the exiting 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 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.)

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 derive 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.

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.

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 project and go to the Solution Explorer window.

2. Click the **Show All Files** button on the top of the window.
3. Find the licenses.licx file and open it. If prompted, continue to open the file.
4. Change the version number of each component to the appropriate value. If the component does not appear in the file, obtain the appropriate data from another licenses.licx file or follow the alternate procedure following.
5. Save the file, then close the licenses.licx tab.
6. Rebuild the project using the **Rebuild All** option (not just **Rebuild**).

Alternatively, follow these steps:

1. Open the project and go to the Solution Explorer window.
2. Click the **Show All Files** button on the top of the window.
3. Find the licenses.licx file and delete it.
4. Close the project and reopen it.
5. Open the main form and add an instance of each licensed control.
6. Check the Solution Explorer window, there should be a licenses.licx file there.
7. Rebuild the project using the **Rebuild All** option (not just **Rebuild**).

For ASP.NET AJAX 2.x applications, follow these steps:

1. Open the project and go to the Solution Explorer window.
2. Find the licenses.licx file and right-click it.
3. Select the Rebuild Licenses option (this will rebuild the App_Licenses.licx file).
4. Rebuild the project using the **Rebuild All** option (not just **Rebuild**).

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 AJAX 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/Support.

Some methods for obtaining technical support include:

- **Online Support via HelpCentral**
  ComponentOne HelpCentral provides customers with a comprehensive set of technical resources in the form of FAQs, samples, Version Release History, Articles, 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 will 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.
- **Peer-to-Peer Product Forums and Newsgroups**
  ComponentOne peer-to-peer product forums and newsgroups are available to exchange information, tips, and techniques regarding ComponentOne products. ComponentOne sponsors these areas as a forum for users to share information. While ComponentOne does not provide direct support in the forums and newsgroups, we periodically monitor them to ensure accuracy of information and provide comments when appropriate. 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**
  ComponentOne documentation is installed with each of our products and is also available online at HelpCentral. 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 AJAX 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.UI.2.dll
- C1.Web.UI.Controls.2.dll

- C1.Web.UI.3.dll
- C1.Web.UI.Controls.3.dll
- C1.Web.UI.4.dll
- C1.Web.UI.Design.4.dll
- C1.Web.UI.Controls.4.dll

Site licenses are available for groups of multiple developers. Please contact Sales@ComponentOne.com for details.

About This Documentation

Acknowledgements

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 ProgressBar As C1.Web.UI.Controls.C1ProgressBar
```

- C#
```
C1.Web.UI.Controls.C1ProgressBar ProgressBar;
```

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
Imports MyProgressBar = MyProject.Objects.C1ProgressBar

Dim wm1 As C1ProgressBar
Dim wm2 As MyProgressBarMenu
```

- C#

```
using C1ProgressBar = C1.Web.UI.Controls.C1ProgressBar;
using MyProgressBar = MyProject.Objects.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 AJAX-Enabled ASP.NET Project

ComponentOne ProgressBar for ASP.NET AJAX requires you to create an ASP.NET AJAX-Enabled project so that Microsoft ASP.NET AJAX Extensions and a **ScriptManager** control are included in your project before the C1ProgressBar control is placed on the page. This allows you to take advantage of ASP.NET AJAX and certain features such as partial-page rendering and client-script functionality of the Microsoft AJAX Library.

When creating AJAX-Enabled ASP.NET projects, Visual Studios 2008 and 2005 both give you the option of creating a Web site project or a Web application project. MSDN provides detailed information on why you would choose one option over the other.

If you are using Visual Studio 2008 with .NET Framework 2.0 or .NET Framework 3.0 or if you are using Visual Studio 2005, you must install the ASP.NET AJAX Extensions 1.0, which can be found at http://ajax.asp.net/. Additionally for Visual Studio 2005 users, creating a Web application project requires installation of a Visual Studio 2005 update and add-in, which can be found at http://msdn.microsoft.com/; however, if you have Visual Studio 2005 SP1, Web application project support is included and a separate download is not required.

If you are using Visual Studio 2008 and .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.

Note: If you are using Visual Studio 2010, see http://www.asp.net/ajax/ for more information on creating an AJAX-Enabled ASP.NET Project.

The following table summarizes the installations needed:

Visual Studio Version	Additional Installation Requirements
Visual Studio 2008, .NET Framework 3.5	None

Visual Studio 2008 and .NET Framework 2.0 or 3.0 Visual Studio 2005 Service Pack 1	ASP.NET AJAX Extensions 1.0 http://www.asp.net/ajax/downloads/archive/
Visual Studio 2005	ASP.NET AJAX Extensions 1.0 Visual Studio update and add-in (2 installs for Web application project support)

The following topics explain how to create both types of projects in Visual Studio 2008 and 2005.

- Creating an AJAX-Enabled Web Site Project in Visual Studio 2008

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

  1. From the File menu, select **New** | Web Site. The New Web Site dialog box opens.
  2. Select .NET Framework 3.5 or the desired framework in the upper right corner. Note that if you choose .NET Framework 2.0 or 3.0, you must install the extensions first.
  3. In the list of templates, select **AJAX 1.0-Enabled ASP.NET 2.0 Web Site**.
  4. Click **Browse** to 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 AJAX-Enabled Web Site is created at the root of the Web server you specified. In addition, a new Web Forms page called Default.aspx is displayed and a **ScriptManager** control is placed on the form. The **ScriptManger** is needed to enable certain features of ASP.NET AJAX such as partial-page rendering, client-script functionality of the Microsoft AJAX Library, and Web-service calls.

- Creating an AJAX-Enabled Web Application Project in Visual Studio 2008

  To create a new Web application project in Visual Studio 2008, complete the following steps.

  1. From the **File** menu, select **New** | Project. The New Project dialog box opens.
  2. Select .NET Framework 3.5 or the desired framework in the upper right corner. Note that if you choose .NET Framework 2.0 or 3.0, you must install the extensions first.
  3. 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**.
  4. Select **AJAX 1.0-Enabled ASP.NET 2.0 Web Application** from the list of **Templates** in the right pane.
  5. Enter a URL for your application in the **Location** field and 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 Forms project is created at the root of the Web server you specified. In addition, a new Web Forms page called Default.aspx is displayed and a **ScriptManager** control is placed on the form. The **ScriptManger** is needed to enable certain features of ASP.NET AJAX such as partial-page rendering, client-script functionality of the Microsoft AJAX Library, and Web-service calls.

- Creating an AJAX-Enabled Web Site Project in Visual Studio 2005

To create a Web site project in Visual Studio 2005, complete the following steps:

1. From the **File** menu in Microsoft Visual Studio .NET, select **New Web Site**. The **New Web Site** dialog box opens.
2. Select **ASP.NET AJAX-Enabled Web Site** from the list of Templates.
3. Enter a URL for your site in the **Location** field and 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 Forms project is created at the root of the Web server you specified. In addition, a new Web Forms page called Default.aspx is displayed and a **ScriptManager** control is placed on the form. The **ScriptManger** is needed to enable certain features of ASP.NET AJAX such as partial-page rendering, client-script functionality of the Microsoft AJAX Library, and Web-service calls.

- Creating an AJAX-Enabled Web Application Project in Visual Studio 2005

To create a new Web application project in Visual Studio 2005, complete the following steps.

1. From the **File** menu in Microsoft Visual Studio 2005, select **New Project**. The **New Project** dialog box opens.
2. Under **Project Types**, choose either **Visual Basic Projects** or **Visual C# Projects**. Note that one of these options may be located under **Other Languages**.
3. Select **ASP.NET AJAX-Enabled Web Application** from the list of **Templates** in the right pane.
4. Enter a URL for your application in the **Location** field and 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 Forms project is created at the root of the Web server you specified. In addition, a new Web Forms page called Default.aspx is displayed and a **ScriptManager** control is placed on the form. The **ScriptManger** is needed to enable certain features of ASP.NET AJAX such as partial-page rendering, client-script functionality of the Microsoft AJAX Library, and Web-service calls.

Adding the ASP.NET Components to a Project

When you install ComponentOne Studio for ASP.NET AJAX, the **Create a ComponentOne Visual Studio Toolbox Tab** check box is checked, by default, in the installation wizard. When you open Visual Studio 2005, you will notice a ComponentOne Studio for ASP.NET AJAX Projects tab containing the ComponentOne controls that have automatically been added to the Toolbox.

If you decide to uncheck the Create a ComponentOne Visual Studio 2005 Toolbox Tab check box during installation, you can manually add ComponentOne controls to the Toolbox at a later time.

Manually Adding the Studio for ASP.NET AJAX controls to the Toolbox
When you install **ComponentOne Studio for ASP.NET AJAX,** the C1ProgressBar component will appear in the Visual Studio Toolbox.

To manually add the Studio for ASP.NET AJAX 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 AJAX 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 AJAX, 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.UI.Controls.2.dll. 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 AJAX Controls to the Form

To add Studio for ASP.NET AJAX 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.UI.Controls.2 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 AJAX** assembly from the list on the **NET** tab or browse to find the C1.Web.UI.Controls.2.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.UI.Controls
```

> **Note:** This makes the objects defined in the **C1.Web.UI.Controls.2** assembly visible to the project. See Namespaces (page 11) for more information.

Key Features

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

- **31 Built-In Animations**

  Choose from 31 built-in animations to spice up the progress bar's fill effect. You can choose how fast and how frequently each animation runs. See C1ProgressBar Animation Effects (page 26) for more information.
- **Professional Visual Styles**

  Change C1ProgressBar''s visual style to one of 5 professional styles by setting one property. And if one of those styles doesn't suit your needs, you can easily customize each of our visual styles using the visual style designer that comes as part of the **Studio for ASP.NET AJAX** package. See Visual Styles (page 25) for more information.
- **Modifiable ToolTips**

Create a more user-friendly Web application by adding a ToolTip to the progress bar. ToolTips 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 ToolTips (page 28) for more information.

- **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 24) for more information.

- **Rich client-side object model**

  Make your Web applications more efficient by using **C1ProgressBar**'s rich client-side object model. C1ProgressBar provides client-side properties, methods, and events.

- **Browser support**

  C1ProgressBar includes browser support for Netscape, Internet Explorer 6, Internet Explorer 7, and Safari.

- **Section 508 compliance**

  The C1ProgressBar control meets Section 508 federal regulations.

- **XHTML 1.1 Compliance**

  C1ProgressBar provides complete XHTML compliance. The output that is generated is fully XHTML 1.1 compliant.

- **Search Engine Optimization**

  Optimized for search engines, C1ProgressBar uses semantic lists and **<a>** tags which are recognized and indexed by Web crawlers.

ProgressBar for ASP.NET AJAX Quick Start

In this **ProgressBar for ASP.NET AJAX** quick start, you will learn how to use a C1ProgressBar in conjunction with an AJAX **Timer** control. The **Timer** control will throw a **Tick** event every second; each time the **Tick** event is thrown, the progress bar's status indicator will leap forward by five percent. When the status bar fills, the **Timer** control will be disabled and the progress bar's label will display a "Task Completed!" message.

To begin this quick start, start with the steps in Step 1 of 4: Creating the Project and Adding Controls (page 16).

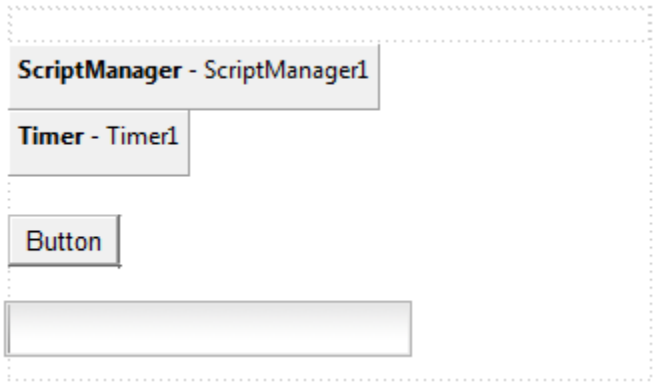
Step 1 of 4: Creating the Project and Adding Controls

In this step of the quick start, you will create your ASP.NET Web site project and then add a **ScriptManager** control, a **Timer** control, a standard **Button** control, and a C1ProgressBar component to it. The **ScriptManager** is essential because ASP.NET AJAX cannot be utilized without it; the other three controls, the **Timer** control, **Button** control, and C1ProgressBar control, will be used in conjunction with one another to reach the final goal of this quick start.

1. Create a new ASP.NET Web site.
2. Click the **Design** button to enter Design view.
3. In the Visual Studio Toolbox, complete the following:

- Double-click the **ScriptManager** icon to add the control to your project.
- Double-click click the **Timer** icon to add the control to your project.
- Double-click the **Button** icon to add the control to your project.
- Double-click the C1ProgressBar icon to add the control to your project.

4. In the Design pane, place your cursor after the Button control and hit enter twice to create breaks between the Button control and the C1ProgressBar control. Your project will resemble the following image:



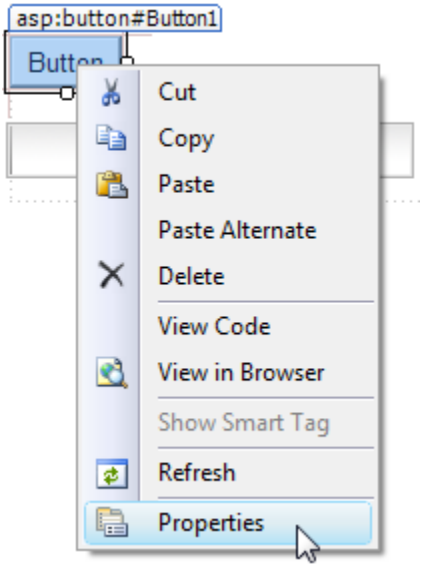
Step 1 of 4 Completed

In this step, you created an ASP.NET Web site and enabled AJAX by adding a **ScriptManager** control. You also added three additional controls, **Button**, **Timer**, and C1ProgressBar, to the project. In the next step, you will configure the **Button**, **Timer**, and C1ProgressBar control's by setting their properties.

Step 2 of 4: Configuring the Controls

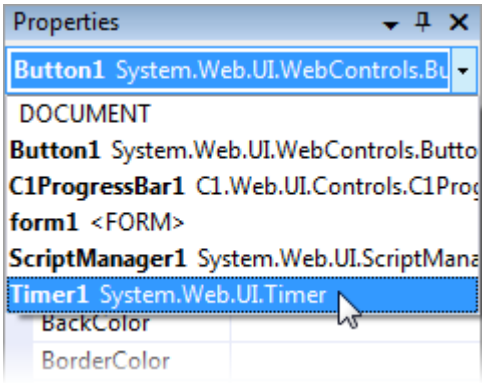
In this step of the quick start, you will set properties of the **Timer** control, the **Button** control, and the C1ProgressBar control. You will modify the duration between the **Timer**'s **Tick** events; you will set the **Button**'s **Text** property; and you will set properties on the C1ProgressBar control that will modify its appearance and behavior.

1. Right-click the **Button** control to open its context menu and select **Properties** from the list.



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

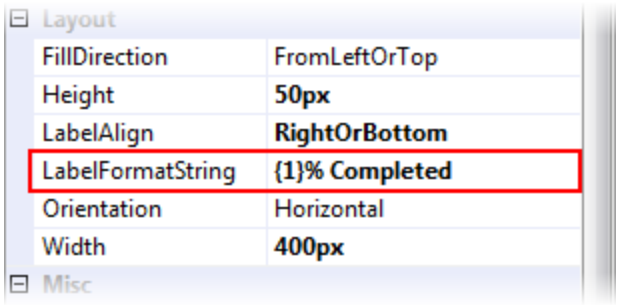
2. Set the **Button** control's **Text** property to "Start Timer".
3. Click the Properties window drop-down arrow and select **Timer1** from the list.



The **Timer** control's properties take focus in the Properties window.

4. Set the **Timer** control's properties as follows:
   - Set the **Interval** property to "1000". This will cause the **Timer** control's **Tick** event to be thrown every 1000 milliseconds (1 second).
   - Set the **Enabled** property to **False**. This will disable the **Timer** control at run time.
5. Click the **Properties** window drop-down arrow and select **C1ProgressBar1**.
6. The C1ProgressBar control's properties take focus in the Properties window.
7. Set the C1ProgressBar control's properties as follows:
   - Set the Height property to "50" to increase the height of the control.
   - Set the Width property to "400" to increase the width of the control.

- Set the AnimationDuration property to "0" to turn off the control's animation effects.
- Set the LabelAlign property to **RightorBottom** to align the label to the right side of the control.
- Set the LabelFormatString property to "{1}% Completed" so that the completed progress is shown on the control's label at run time.



Step 2 of 4 Completed

In this step, you configured the **Button**, **Timer**, and C1ProgressBar controls by setting their properties. In the next step, you will create two event handlers, **Tick** event for the **Timer** control and a **Click** event for the **Button** control, and then add behavior-modifying code to the each event handler.

Step 3 of 4: Coding the Project

In this step of the quick start, you will create a **Timer1_Tick** event handler and a **Button1_Click** event handler. In the **Timer1_Tick** event handler, you will add code that determines the behavior of the C1ProgressBar control; in the **Button_Click** event handler, you will add code that initializes the **Timer** control and resets the Value property of the C1ProgressBar.

1. Double-click the **Timer** control. This will add the **Timer1_Tick** even to Code view.
2. Place the following code within the **Timer1_Tick** event:

- Visual Basic

```
If C1ProgressBar1.Value = C1ProgressBar1.MaximumValue Then
'When the progress bar reaches MaximumValue, stop the timer and change
label value / alignment
   Timer1.Enabled = False
   C1ProgressBar1.LabelFormatString = "Task Completed!"
   C1ProgressBar1.LabelAlign =
C1.Web.UI.Controls.C1ProgressBar.ProgressBarLabelAlign.Center
End If
If True Then
   'If C1ProgressBar's Value property is less than the MaximumValue
property, increase the Value property by 5
   C1ProgressBar1.Value = C1ProgressBar1.Value + 5
End If
```

- C#

```
if (C1ProgressBar1.Value == C1ProgressBar1.MaximumValue)
{
   //When the progress bar reaches MaximumValue, stop the timer and
change label value/alignment
   Timer1.Enabled = false;
```

```
        C1ProgressBar1.LabelFormatString = "Task Completed!";
        C1ProgressBar1.LabelAlign = ProgressBarLabelAlign.Center;
    }
    {
        // If C1ProgressBar's Value property is less than the MaximumValue
    property, increase the Value property by 5
        C1ProgressBar1.Value = C1ProgressBar1.Value + 5;
    }
```

3. Click the proper **Default.aspx** tab to return to Design view.
4. Double-click the **Button** control to add the **Button1_Click** event handler to code view.
5. Place the following code within the **Button1_Click** event:

- Visual Basic

```
'Enable Timer control and reset C1ProgressBar.Value upon button click
Timer1.Enabled = true
C1ProgressBar1.Value = 0
```

- C#

```
//Enable Timer control and reset C1ProgressBar.Value upon button click
Timer1.Enabled = true;
C1ProgressBar1.Value = 0;
```

Step 3 of 4 Completed

In this step, you coded behaviors for the **Timer** control and the C1ProgressBar control. In the next step, you will run the project and observe the behaviors.

Step 4 of 4: Running the Project

In this step, you will build the Web project and observe the behaviors that we added to the C1ProgressBar control.

1. Press **F5** to run the project.
2. Click **Start Timer** to begin the **Timer** control. Observe that the progress bar increases by 5% every second and that the label is on the right side of the control.
3. Wait for the C1ProgressBar control to fill. When it fills, observes that progress bar's label aligns to the center of the control and that the string now reads as "Task Completed!"

Step 4 of 4 Completed

Congratulations! You have completed the **ProgressBar for ASP.NET AJAX** Quick Start. Now that you've completed the tutorial, you may want to go back and play around with some of the settings to see how you can achieve different results using this control.

ProgressBar for ASP.NET AJAX Top Tips

This section provides tips and tricks for using the C1ProgressBar control.

- Remember to always use the **ScriptManager** control.
- Set visual styles on your **Studio for ASP.NET AJAX** control to add rich themes to your application. For task-based help, see Changing the Visual Style (page 38). For a list of visual styles, see Visual Styles (page 25).
- Use image sprites in your custom visual styles to increase performance and decrease load times.
- Use the built-in animations to add transitional effects to your Web site or Web application.
- Add easing in and easing out transitions to make your animations smoother and more natural.See C1ProgressBar Animations (page 26) for a list of animation effects. To learn how to add an animation effect to the C1ProgressBar control, see Configuring C1ProgressBar Animations (page 40).
- Extend our animation library to *any* part of your web application, including portions that aren't ComponentOne controls. See C1ProgressBar Animations (page 26) for a complete list of animation and transition effects.
- Update the client-side model properties and events when you don't need to perform server-side processing. The C1ProgressBar control can be coded on both the client side and server side.
- You can use the **C1ProgressBarTaskEventArgs.UpdateProgress** method to associate the C1ProgressBar control with a long running server-side task.

  For example:

  - ASPX Markup
  ```
  <cc1:C1ProgressBar runat="server" ID="C1ProgressBar1"
  onruntask="C1ProgressBar1_RunTask"
              StartTaskButton="Button1" StopTaskButton="Button2"
  ></cc1:C1ProgressBar>
  <asp:Button ID="Button1" runat="server" Text="Start" />
  <asp:Button ID="Button2" runat="server" Text="Stop" />
  ```
  - C#
  ```
      protected void C1ProgressBar1_RunTask(object sender,
  C1ProgressBarTaskEventArgs e)
      {
          for (int i = 1; i <= 100; i++)
          {
              System.Threading.Thread.Sleep(1000);
              e.UpdateProgress(i);
          }
      }
  ```

  When you click **Button1**, the `C1ProgressBar1_RunTask` method starts, and when you click **Button2**, the `C1ProgressBar1_RunTask` method stops working.

  While the progress bar is running, you cannot update any other UI elements on the page through traditional means because nothing on the page is being loaded while the project bar is updated (A traditional update occurs when the page is reloaded, which is not the case for a progress bar).

  C1ProgressBarTaskEventArgs.UpdateProgress provides a second parameter for you to pass additional data to the client, which you can use together with JavaScript to update client UI elements when the progress bar is running.

C1ProgressBar Design-Time Support

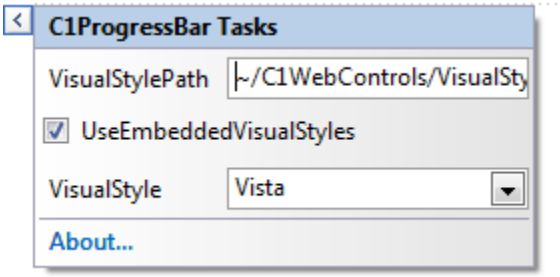
C1ProgressBar provides smart tags and a designer that offers rich design-time support and simplifies working with the object model.

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

C1ProgressBar Smart Tag

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:

- **Visual Style Path**

  Enter the path to your custom visual style in this text box.

- **UseEmbedded Visual Styles**

  Select this box to use one of the five built-in visual styles. If you want to use a custom visual style, deselect this box.

- **VisualStyle**

  Add a visual style for the C1ProgressBar control by selecting a theme from the **VisualStyle** drop-down list. You can choose from **ArcticFox**, **Office2007Black**, **Office2007Blue**, **Office2007Silver**, and **Vista**.

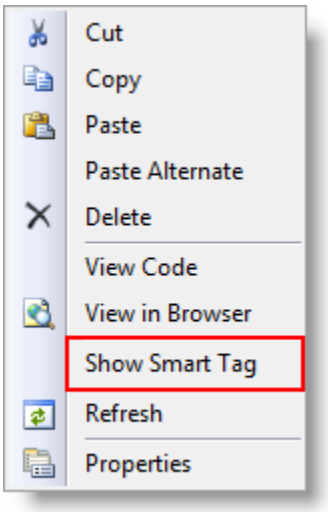
- **About**

  Clicking **About** reveals the **About ComponentOne** dialog box. This dialog box displays the version number and licensing information for the ComponentOne 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:

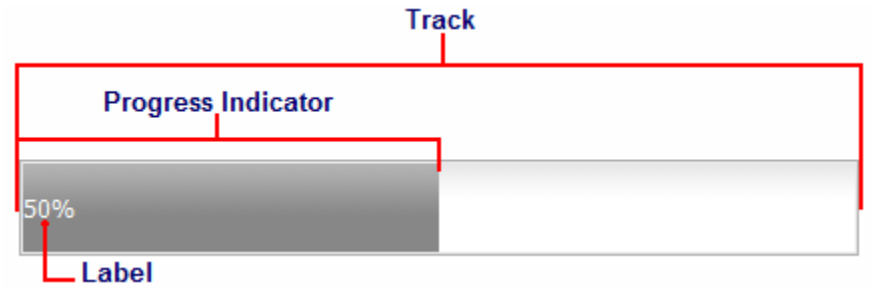
- **Show Smart Tag**

  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. If the C1ProgressBar control's track is horizontal, the indicator can fill the track starting from either the left or the right side of the control; if the track is vertical, the indicator can fill the track starting from either the top or the bottom 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 MinimumValue and MaximumValue properties. For example, if the MinimumValue property is set to **0**, the MaximumValue is set to **100**, and the Value property is set to **25**, the progress indicator will fill 25% of the track. If the MinimumValue property is set to **100**, the MaximumValue 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 MinimumValue property.
{*5*} or {*Max*}	Displays the value of the MaximumValue 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.

The label element can be aligned in several different ways. If the C1ProgressBar is horizontal, it can be aligned to the left, right, or center, or it can run as a marquee that travels ahead of the progress bar; if the C1ProgressBar control is vertical, it can be aligned to the top, center, or bottom of the control, or it can be run as a marquee that travels ahead of the progress bar.

C1ProgressBar Appearance

The following topics provide information about the appearance of the C1ProgressBar control.

Visual Styles

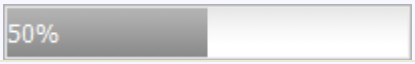
The C1ProgressBar control provides five built-in visual styles that can be easily applied using the VisualStyle property. You can choose from one of the following schemes: **ArcticFox** (default), **Office2007Black**, **Office2007Blue**, **Office2007Silver**, and **Vista**.

Using the Visual Styles

To learn how to add a preformatted scheme to C1ProgressBar, see Changing the Visual Style (page 38).

Appearance of the Visual Styles

The five visual styles appear as follows:

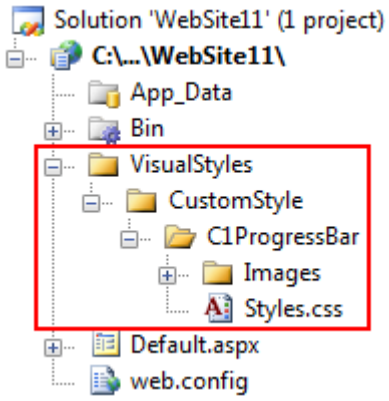
Format	Appearance
ArcticFox	
Office200 Black	
Office2007Blue	
Office2007Silver	
Vista	

Custom Visual Styles

While **ProgressBar for ASP.NET AJAX** comes with five built-in styles, we recognize that there are instances where you may want to customize your C1ProgressBar control. To customize the C1ProgressBar control, you will create a custom CSS style sheet and add it to your project as a visual style. The custom CSS style sheet must always be named "styles.css".

Tip: The easiest way to create a custom visual style is by modifying one of the control's pre-existing visual styles. You can find the .css style sheets and images for **C1ProgressBar**'s visual styles within the installation directory at *C:\Program Files\ComponentOne\Studio for ASP.NET AJAX\C1WebUI\VisualStyles.*

Before you add your .css file and images, you will have to create a hierarchy of folders, the last of which will contain your files. On the top-level of your project, create a folder named "VisualStyles". Underneath the VisualStyles folder, create a sub-folder bearing the theme name (such as "CustomStyle"), and then, beneath that, create a sub-folder named "C1ProgressBar". The image folder and .css file should be placed underneath the **C1ProgressBar** folder. The result will resemble the following image:



This structure of these folders is very important; **C1ProgressBar** will always look for the **~/VisualStyles/StyleName/C1ProgressBar/styles.css** path, as it is the default visual style path for the control.

Once the .css file and images are in place, set the **VisualStylePath** property to the path of the first folder (~/VisualStyles), set the **UseEmbeddedVisualStyles** property to **False,** and then set the VisualStyle property to the custom theme name.

C1ProgressBar Behavior

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

C1ProgressBar Animations

The C1ProgressBar control features 31 animation transition effects that can be used to modify the way the progress indicator fills in the progress bar track. The following topics provide information about C1ProgressBar"s animation effects.

C1ProgressBar Animation Effects

The C1ProgressBar control contains thirty-one built-in animation transition effects that allow you to customize the fill effect of the progress bar's progress indicator. The default transition is **EaseLinear**, but you can set it to another transitional effect by setting the **Easing** property.

The table below describes each animation effect:

Name	Description
EaseLinear (default)	Linear easing. Moves smoothly without acceleration or deceleration.
EaseOutElastic	Elastic easing out. Starts quickly and then decelerates.
EaseInElastic	Elastic easing in. Starts slowly and then accelerates.
EaseInOutElastic	Elastic easing in and out. Starts slowly, accelerates, and then decelerates.
EaseOutBounce	Bouncing easing out. Starts quickly and then decelerates.
EaseInBounce	Bouncing easing in. Starts slowly and then accelerates.
EaseInOutBounce	Bouncing easing in and out. Starts slowly, accelerates, and then decelerates.
EaseOutExpo	Exponential easing out. Starts quickly and then decelerates.
EaseInExpo	Exponential easing in. Starts slowly and then accelerates.
EaseInOutExpo	Exponential easing in and out. Starts slowly, accelerates, and then decelerates.
EaseOutQuad	Quadratic easing out. Starts quickly and then decelerates.
EaseInQuad	Quadratic easing in. Starts slowly and then accelerates.
EaseInOutQuad	Quadratic easing in and out. Starts slowly, accelerates, and then decelerates.
EaseOutSine	Sinusoidal easing out. Starts quickly and then decelerates.
EaseInSine	Sinusoidal easing in. Starts slowly and then accelerates.
EaseInOutSine	Sinusoidal easing in and out. Starts slowly, accelerates, and then decelerates.
EaseOutCirc	Circular easing out. Starts quickly and then decelerates.
EaseInCirc	Circular easing in. Starts slowly and then accelerates.
EaseInOutCirc	Circular easing in and out. Starts slowly, accelerates, and then decelerates.
EaseOutCubic	Cubic easing out. Starts quickly and then decelerates.
EaseInCubic	Cubic easing in. Starts slowly and then accelerates.
EaseInOutCubic	Cubic easing in and out. Starts slowly, accelerates, and then decelerates.
EaseOutQuint	Quintic easing out. Starts quickly and then decelerates.
EaseInQuint	Quintic easing in. Starts slowly and then accelerates.
EaseInOutQuint	Quintic easing in and out. Starts slowly, accelerates, and then decelerates.
EaseOutBack	Back easing out. Starts quickly and then decelerates.
EaseInBack	Back easing in. Starts slowly and then accelerates.
EaseInOutBack	Back easing in and out. Starts slowly, accelerates, and then decelerates.
EaseOutQuart	Quartic easing out. Starts quickly and then decelerates.
EaseInQuart	Quartic easing in. Starts slowly and then accelerates.
EaseInOutQuart	Quartic easing in and out. Starts slowly, accelerates, and then decelerates.

Animation Effect Duration

You can set the length of C1ProgressBar"s animation effect using the AnimationDuration property. The unit of time used for specifying animation effect duration is in milliseconds, and the default setting for the AnimationDuration property is **100** milliseconds (or half a second). Increase this value for a longer animation effect, and decrease this number for a shorter animation effect.

Animation Effect Delay

You can specify the length of time that has to pass before the progress bar's animation effect begins by setting the AnimationDelay property. The unit of time used for specifying animation effect duration is in milliseconds, and the default setting for the AnimationDuration property is **0** milliseconds, which will cause the animation to begin immediately. Increase this value to extend the period before an animation is fired, and decrease this number to shorten the period before an animation is fired.

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 MinimumValue property.
{*5*} or {*Max*}	Displays the value of the MaximumValue property.

Client-Side Functionality

C1ProgressBar includes a rich and flexible client-side object model. Several server-side properties and methods can be used on the client-side. Client-side events can be accessed from the Properties window.

When a C1ProgressBar control is added to a web project, an instance of the client-side tab control will be created automatically. For example, if you have a **C1ProgressBar** control with the server-side ID of "C1ProgressBar1", you can use the following script to acquire a reference to its client object:

```
var NewProgressBar = $find("<%= C1ProgressBar1.ClientID %>");
```

Using C1ProgressBar''s client-side functionality, you can implement many features in your Web page without having to post back to the server. Thus, using client-side methods and properties will increase the efficiency of your Web site.

The following topics describe the available client-side properties, methods, and events.

Client-Side Properties

The following conventions are used when accessing the client object properties:

- Server properties on the client are implemented as a pair of Get- and Set- methods.
- Method names must start with "get_" (Get-method) and "set_" (Set-method) followed with the server property name. The first letter of the server property name must be lowercase (camel case).

The following JavaScript example sets the selected index of a C1ProgressBar control to **50**.

```
<script id="ScriptSample" type="text/javascript">
function SetMaximum ()
{
  $find("<%=C1ProgressBar1.ClientID%>").set_maximumValue(50);
};
</script>
```

For more information on client-side properties, see **C1ProgressBar**'s client-side reference.

Client-Side Methods

C1ProgressBar includes a rich client-side object model that includes several client-side methods. For information about these client-side methods, see the **C1ProgressBar**'s client-side reference.

Client-Side Events

C1ProgressBar includes a rich client-side object model in which includes several client-side events. You can access these client-side events from the Properties window. To create a new client-side event, select the drop-down arrow next to a client-side event and select **Add new client side handler**.
Once you've added the client-side handler, Visual Studio will add a script to the Source view. That script will resemble the following:

```
<script id="ComponentOneClientScript" type="text/javascript">
function C1ProgressBar1_OnClientMouseOver(){
     //
     // Put your code here.
     //
};
</script>
```

The following provides names and descriptions of C1ProgressBar''s client-side events:

Client-Side Event	Description
OnClientBeforeProgressChanging	Raised before the value of the progress bar changes.
OnClientClick	Raised when the progress bar has been clicked.
OnClientMouseDown	Raised when the mouse is down on the progress bar.

OnClientMouseOut	Raised when the mouse leaves the progress bar.
OnClientMouseOver	Raised then the mouse is over the progress bar.
OnClientMouseUp	Raised when the mouse is up on the progress bar.
OnClientProgressChanged	Raised when the value of the progress bar has been changed.
OnClientProgressChanging	Raised when the value of the progress bar is changing.

ProgressBar for ASP.NET AJAX Samples

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

You can access samples from the **ComponentOne Control Explorer**. To view samples, click the **Start** button and then click **ComponentOne | Studio for ASP.NET | Control Explorer.** The following table provides a short overview of each sample.

Sample	Description
Visual Styles	Illustrates the **C1ProgressBar** control's five built-in visual styles.
Client Side	Illustrates several client-side property settings of the **C1ProgressBar** control.

ProgressBar for ASP.NET AJAX 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 AJAX-enabled ASP.NET project. **ComponentOne ProgressBar for ASP.NET AJAX** requires you to create an ASP.NET AJAX-Enabled project so that Microsoft ASP.NET AJAX Extensions and a **ScriptManager** control are included in your project before the C1ProgressBar control is placed on the page. This allows you to take advantage of ASP.NET AJAX and certain features such as partial-page rendering and client-script functionality of the Microsoft AJAX Library.

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 to the left (if horizontal) or to the top (if vertical) of the control. In this topic, you will learn how to change the alignment of the label in Design view, in Source view, and

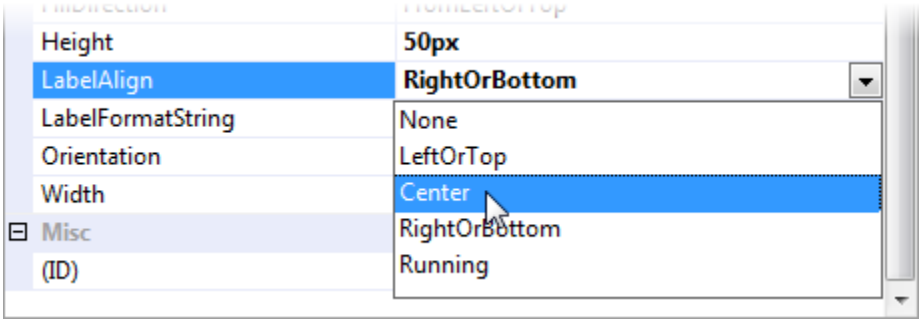
in code. This topic assumes that you have created an ASP.NET AJAX-Enabled Web site containing a **ScriptManager** control and a C1ProgressBar control.

Aligning the Label in Design View

To align the label, 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 **Center** from the list.



4. Press F5 to run the project and observe that the label now appears in the center of the control.

Aligning the Label in Source View

To align the label, complete the following steps:

1. Click the **Source** button to enter Source view.
2. Add `LabelAlign = "Center"` to the `<cc1:C1ProgressBar>` tag so that the markup resembles the following:

```
<cc1:C1ProgressBar ID="C1ProgressBar2" runat="server" LabelAlign="Center"
/>
```

3. Press **F5** to run the project and observe that the label now appears in the center of the control.

Aligning the Label in Code

To align the label, 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.UI.Controls.C1ProgressBar
     ```

   - C#

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

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

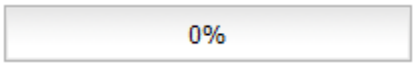
- Visual Studio
```
C1ProgressBar1.LabelAlign = ProgressBarLabelAlign.Center
```

- C#
```
C1ProgressBar1.LabelAlign = ProgressBarLabelAlign.Center;
```

4. Press **F5** to run the project and observe that the label now appears in the center of the control.

This Topic Illustrates the Following:

In this topic, you aligned the C1ProgressBar control's label to the center. 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

To format the label, 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 build 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

To format the label, complete the following steps:

1. Click the **Source** button to enter Source view.
2. Place `LabelFormatString = "Maximum Value: {5}"` in the `<cc1:C1ProgressBar>` tag so that the markup resembles the following:

```
<cc1:C1ProgressBar ID="C1ProgressBar2" runat="server"
Width="288px" Height="22px" UseEmbeddedVisualStyles="True"
VisualStyle="ArcticFox" LabelFormatString="Maximum value: {5}" />
```

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

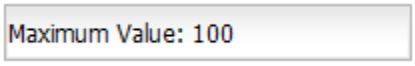
Formatting the Label in Code

To format the label, 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.UI.Controls.C1ProgressBar
     ```
   - C#
     ```
     using C1.Web.UI.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 result of this topic resembles the following:



Formatting the C1ProgressBar ToolTip

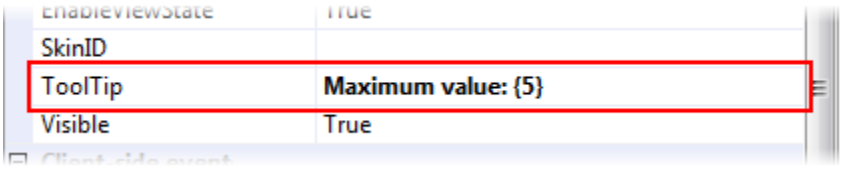
At run time, the C1ProgressBar control will display a ToolTip if a user runs his cursor over it. By default, this ToolTip will display the current percentage of progress. In this topic, you will customize the ToolTip so that it displays a modified string along with the maximum value setting of the control. 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

To format the ToolTip, 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 ToolTip property to "Maximum Value: {5}".



4. Press **F5** to build 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

To format the ToolTip, complete the following steps:

1. Click the **Source** button to enter Source view.
2. Place `ToolTip = "Maximum Value: {5}"` in the `<cc1:C1ProgressBar>` tag so that the markup resembles the following:

```
<cc1:C1ProgressBar ID="C1ProgressBar2" runat="server"
Width="288px" Height="22px" UseEmbeddedVisualStyles="True"
VisualStyle="ArcticFox" ToolTip="Maximum value: {5}" />
```

3. 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

To format the ToolTip, 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.UI.Controls.C1ProgressBar
     ```
   - C#
     ```
     using C1.Web.UI.Controls.C1ProgressBar;
     ```
3. 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}";
     ```
4. 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 result of this topic resembles the following:

Changing the C1ProgressBar Orientation

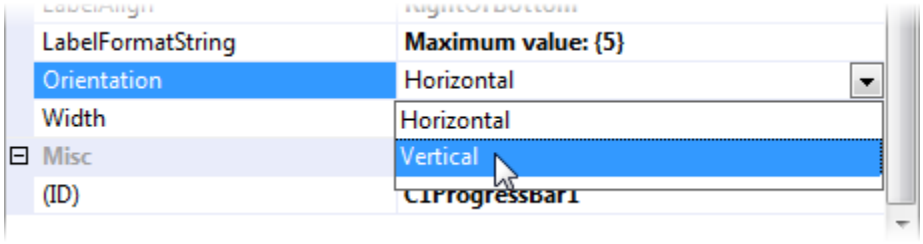
By default, the orientation of the C1ProgressBar control is horizontal, meaning that the progress indicator will fill from either the right or the left of the control. However, you can set the progress indicator to fill from the top or the bottom of the control by setting the Orientation property to Vertical. In this topic, you will set the Orientation property in Design view, in Source view, and in code. This topic assumes that you have created an ASP.NET AJAX-Enabled Web site containing a **ScriptManager** control and a C1ProgressBar control.

Changing the Orientation in Design View

To change the orientation, 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, complete the following:
   - Set the Orientation property to **Vertical**.



> **Note**: Setting the Orientation property to **Vertical** will not reverse the width and height of the control; you will have to do that manually by setting the Width and Height properties.

   - Set the Value property to "100". This will cause the progress indicator to fill the control at run time.
   - Set the AnimationDuration property to "6000". This will lengthen the progress indicator's fill effect so that you won't miss it at run time.
4. Press **F5** to run the project and observe that the progress indicator starts filling the track starting from the top of the control. You can also have it fill from the bottom of the control by changing the setting of the FillDirection property (see Changing the Progress Indicator Fill Direction (page 37)).

Changing the Orientation in Source View

To change the orientation, complete the following steps:

1. Click the **Source** button to enter Source view.
2. Add `Orientation="Vertical"` , `Value="100"`, and `AnimationDuration="6000"` to the `<cc1:C1ProgressBar>` tag so that the markup resembles the following:

```
<cc2:C1ProgressBar ID="C1ProgressBar2" runat="server"
Orientation="Vertical" Value="100" AnimationDuration="6000"/>
```

The other two properties, Value and AnimationDuration, are being set so that you can see the effect of the orientation change at run time. Setting the Value property to **100** will cause the progress indicator to fill, and setting the AnimationDuration property to **6000** will lengthen the animation so that you don't miss the fill effect.

3. Press **F5** to run the project and observe that the progress indicator starts filling the track starting from the top of the control. You can also have it fill from the bottom of the control by changing the setting of the **FillDirection** property (see Changing the Progress Indicator Fill Direction (page 37)).

Changing the Orientation in Code

To change the orientation, 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.UI.Controls.C1ProgressBar
     ```
   - C#
     ```
     using C1.Web.UI.Controls.C1ProgressBar;
     ```
3. Add the following code to the **Page_Load** event:
   - Visual Basic
     ```
     'Set the Orientation property to Vertical
     C1ProgressBar1.Orientation = Orientation.Vertical

     'Lengthen the fill effect duration for run-time observance
     C1ProgressBar1.AnimationDuration = 6000

     'Specify value so progress indicator will fill at run time
     C1ProgressBar1.Value = 100
     ```
   - C#
     ```
     //Set the Orientation property to Vertical
     C1ProgressBar1.Orientation = Orientation.Vertical;

     //Lengthen the fill effect duration for run-time observance
     C1ProgressBar1.AnimationDuration = 6000;

     //Specify value so progress indicator will fill at run time
     C1ProgressBar1.Value = 100;
     ```
4. Press **F5** to run the project and observe that the progress indicator starts filling the track starting from the top of the control. You can also have it fill from the bottom of the control by changing the setting of the **FillDirection** property (see Changing the Progress Indicator Fill Direction (page 37)).

Changing the Progress Indicator Fill Direction

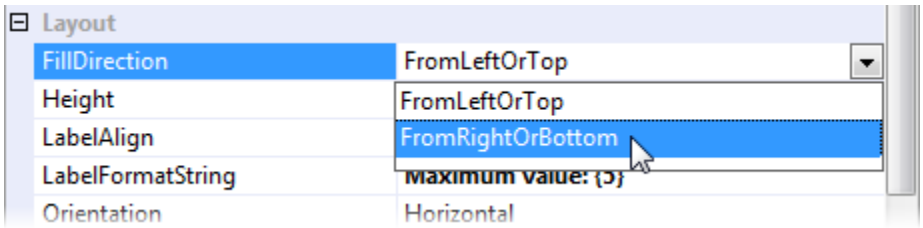
By default, the progress indicator will fill the track starting from the left (if horizontal) or the top (if vertical) of the control. The fill direction can be set by setting one property: **FillDirection**. In this topic, you will set the **FillDirection** property in Design view, in Source view, and in code.

Changing the Fill Direction in Design View

To change the fill direction, 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, complete the following:
   - Set the **FillDirection** property to **FromRightOrBottom**.



   - Set the Value property to "100". This will cause the progress indicator to fill the control at run time.
   - Set the AnimationDuration property to "6000". This will lengthen the progress indicator's fill effect so that you won't miss it at run time.
4. Press **F5** to run the project and observe that the progress indicator starts to fill in the control from the right. If you want it to fill from the bottom, you can change the control's **Orientation** property (see Changing the C1ProgressBar Orientation (page 35)).

Changing the Fill Direction in Source View

To change the fill direction, complete the following steps:

1. Click the **Source** button to enter Source view.
2. Add `FillDirection="FromRightOrBottom"`, `Value="100"`, and `AnimationDuration="6000"` to the `<cc1:C1ProgressBar>` tag so that the markup resembles the following:

```
<cc1:C1ProgressBar ID="C1ProgressBar2" runat="server"
FillDirection = "FromRightOrBottom" Value="100" AnimationDuration="6000"/>
```

   The other two properties, Value and AnimationDuration, are being set so that you can see the effect of the orientation change at run time. Setting the Value property to **100** will cause the progress indicator to fill, and setting the AnimationDuration property to **6000** will lengthen the animation so that you don't miss the fill effect.
3. Press **F5** to run the project and observe that the progress indicator starts to fill in the control from the right. If you want it to fill from the bottom, you can change the control's **Orientation** property (see Changing the C1ProgressBar Orientation (page 35)).

Changing the Fill Direction in Code

To change the fill direction, 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.UI.Controls.C1ProgressBar
     ```
   - C#
     ```
     using C1.Web.UI.Controls.C1ProgressBar;
     ```
3. Add the following code to the **Page_Load** event:
   - Visual Basic
     ```
     'Set the FillDirection property to FromRightOrBottom
     C1ProgressBar1.FillDirection =
     ProgressBarFillDirection.FromRightOrBottom

     'Lengthen the fill effect duration for run-time observance
     C1ProgressBar1.AnimationDuration = 6000

     'Specify value so progress indicator will fill at run time
     C1ProgressBar1.Value = 100
     ```
   - C#
     ```
     // Set the FillDirection property to FromRightOrBottom
     C1ProgressBar1.FillDirection =
     ProgressBarFillDirection.FromRightOrBottom;

     //Lengthen the fill effect duration for run-time observance
     C1ProgressBar1.AnimationDuration = 6000;

     //Specify value so progress indicator will fill at run time
     C1ProgressBar1.Value = 100;
     ```
4. Press **F5** to run the project and observe that the progress indicator starts to fill in the control from the right. If you want it to fill from the bottom, you can change the control's **Orientation** property (see Changing the C1ProgressBar Orientation (page 35)).

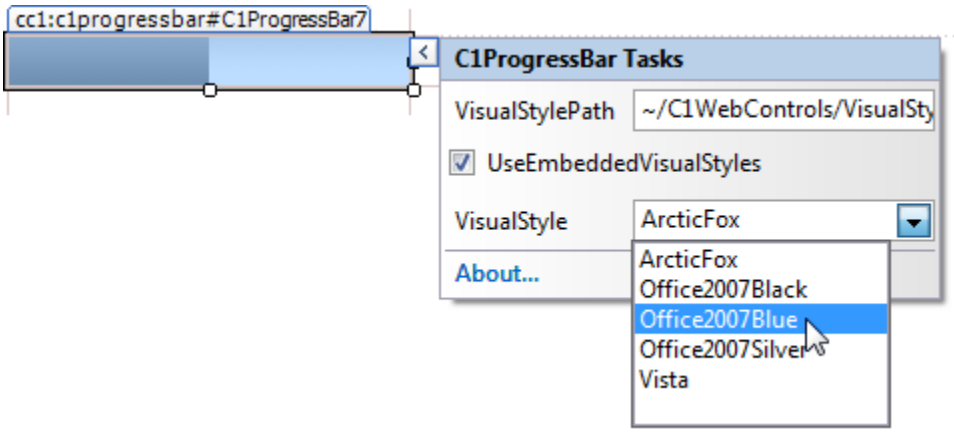
Changing the Visual Style

A C1ProgressBar control has five embedded visual styles that you can apply with just a few clicks. This topic illustrates how to change the visual style in Design view, in Source view, and in code. For more information on visual styles, see Visual Styles (page 25).

Changing the Visual Style in Design View

To change the visual scheme of your **C1ProgressBar**, follow these steps:

1. Click the **C1ProgressBar** smart tag (▶) to open the **C1ProgressBar Tasks** menu.
2. Click the **VisualStyle** drop-down arrow and select a visual style from the list. For this example, select **Office2007Blue**.



The **Office2007Blue** visual scheme is applied to the C1ProgressBar control.

Changing the Visual Style in Source View

To change the visual scheme of your **C1ProgressBar** in Source view, add `VisualStyle="Office2007Blue"` to the `<cc1:C1ProgressBar>` tag so that it resembles the following:

```
<cc1:C1ProgressBar ID="C1ProgressBar1" runat="server"
VisualStyle="Office2007Blue" VisualStylePath="~/C1WebControls/VisualStyles">
```

Changing the Visual Style in Code

To change the visual scheme, follow these steps:

1. Import the following namespace into your project:
   - Visual Basic
     ```
     Imports C1.Web.UI.Controls
     ```
   - C#
     ```
     using C1.Web.UI.Controls;
     ```
2. Add the following code, which sets the VisualStyle property, to the **Page_Load** event:
   - Visual Basic
     ```
     C1ProgressBar1.VisualStyle = "Office2007Blue"
     ```
   - C#
     ```
     C1ProgressBar1.VisualStyle = "Office2007Blue";
     ```
3. Run the program.

This topic illustrates the following:

The following image shows a **C1ProgressBar** control with the **Office2007Blue** visual style:

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

To configure animations, 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 Easing property to **EaseOutBounce**. This establishes the animation effect.
   - Set the AnimationDelay property to "500". This establishes the length of time that will pass before the animation starts.
   - Set the AnimationDuration property to "6000". This establishes the length of the animation.
   - 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

To configure animations, complete the following steps:

1. Click the **Source** button to enter Source view.
2. Add `Easing="EaseOutBounce",AnimationDuraton="6000",AnimationDelay="500",` and `Value = "100"` to the `<cc1:C1ProgressBar>` tag so that the markup resembles the following:

```
<cc2:C1ProgressBar ID="C1ProgressBar2" runat="server"
Value="100" AnimationDuration="40000" AnimationDelay="500"
Easing="EaseOutBounce"/>
```

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.

Configuring Animations in Code

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

```
'Set animation properties
C1ProgressBar1.Easing = C1.Web.UI.Easing.EaseOutBounce
C1ProgressBar1.AnimationDelay = 500
C1ProgressBar1.AnimationDuration = 6000

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

- C#

```
//Set animation properties
C1ProgressBar1.Easing = C1.Web.UI.Easing.EaseOutBounce;
C1ProgressBar1.AnimationDelay = 500;
C1ProgressBar1.AnimationDuration = 6000;

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

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.

Setting C1ProgressBar Values

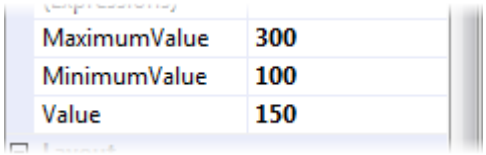
The C1ProgressBar control contains a minimum value, a maximum value, and a value. In this topic, you will change the value of the MinimumValue, MaximumValue, and Value properties from their defaults (0, 100, and 0 respectively) in Design view, in Source view, and in code.

Setting Values in Design View

To set values, 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 MaximumValue property to "300".
   - Set the MinimumValue 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

To set values, complete the following steps:

1. Click the **Source** button to enter Source view.
2. Add `MinimumValue="100"`, `MaximumValue="300"`, and `Value="150"` to the `<cc1:C1ProgressBar>` tag so that the markup resembles the following:

```
<cc2: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

To set values, 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 MaximumValue property, add the following code to the **Page_Load** event:
   - Visual Basic
     ```
     C1ProgressBar1.MaximumValue = 300
     ```
   - C#
     ```
     C1ProgressBar1.MaximumValue = 300;
     ```
4. To set the MinimumValue property, add the following code to the **Page_Load** event:
   - Visual Basic
     ```
     C1ProgressBar1.MinimumValue = 100
     ```
   - C#
     ```
     C1ProgressBar1.MinimumValue = 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).

Client-Side Scripting Tasks

The following topics illustrate how to use several of the C1ProgressBar control's client-side features. For more information on client-side members, see Client-Side Functionality (page 28).

Changing the Fill Direction at Run Time

In this topic, you will write a client-side script that will allow users to change the fill direction of the progress bar at run time. This topic assumes that you have created an ASP.NET AJAX-Enabled Web site containing a **ScriptManager** control and a C1ProgressBar control.

Complete the following:

1. Click the **Source** tab to enter Source view.
2. Place the following markup beneath the `<c1:C1ProgressBar>` tag:

```
<select id="Select1" onchange="changeFillDirection(this.value)"
name="fillDirectionChange">
      <option selected="selected" value="0">FromLeftOrTop</option>
      <option value="1">FromRightOrBottom</option>
</select>
```

3. This markup creates a drop-down list with two options: **FromLeftOrTop** and **FromRightOrBottom**. Each option has been assigned a value, which you will use later to determine the user's input.

   Notice that the **onchange** property of the drop-down list has been set to "changeFillDirection(this.value)". **changeFillDirection** is the name of the function you'll create in the next step. Whenever the selection of the drop-down list has changed, **this.value** (the value of the drop-down list selection) will be passed to the **changeFillDirection** function.
4. Place the following JavaScript after the `<body>` tag:

```
<script type="text/javascript">
function changeFillDirection(val)
    {
var pb = Sys.Application.findComponent("<%=C1ProgressBar1.ClientID%>");
       if(val=="1")
       {
pb.set_fillDirection(C1.Web.UI.Controls.C1ProgressBar.ProgressBarFillDirect
ion.fromRightOrBottom);
       }
       else
       {
pb.set_fillDirection(C1.Web.UI.Controls.C1ProgressBar.ProgressBarFillDirect
ion.fromLeftOrTop);
       }
       }
</script>
```

   This code checks for the value of the selected drop-down list item and then sets the **C1ProgressBar** control's **FillDirection** property to that value.
5. Add `Value="50"` to the `<c1:C1ProgressBar>` tag so that the progress bar will appear half-full at run time.
6. Press F5 to run the program.
7. From the drop-down list, select **FromRightOrBottom**. Observe that the progress bar's status indicator fills from the right.

Changing the Orientation at Run Time

In this topic, you will write a client-side script that will allow users to change the orientation of the progress bar at run time. This topic assumes that you have created an ASP.NET AJAX-Enabled Web site containing a **ScriptManager** control and a C1ProgressBar control.

Complete the following:

1. Click the **Source** tab to enter Source view.
2. Place the following markup beneath the `<c1:C1ProgressBar>` tag:

```
<select id="Select1" onchange="changeOrientation(this.value)"
name="orientationChange">
      <option selected="selected" value="0">Horizontal</option>
      <option value="1">Vertical</option>
</select>
```

   This markup creates a drop-down list with two options: **Horizontal** and **Vertical**. Each option has been assigned a value, which you will use later to determine the user's input.

   Notice that the **onchange** event of the list has been set to "changeOrientation(this.value)". **changeOrientation** is the name of the function you'll create in the next step. Whenever the selection of the drop-down list has changed, **this.value** (the value of the drop-down list selection) will be passed to the **changeOrientation** function.

3. Place the following JavaScript after the `<body>` tag:

```
<script type="text/javascript">
            function changeOrientation(listval)
    {
       var pb =
Sys.Application.findComponent("<%=C1ProgressBar1.ClientID%>");
       if(listval=="1")
       {
            pb.set_orientation(C1.Web.UI.Orientation.vertical);
       }
       else
       {
            pb.set_orientation(C1.Web.UI.Orientation.horizontal);
       }
    }
    </script>
```

   This code checks for the value of the selected drop-down list item and then sets the C1ProgressBar control's **Orientation** property to that value.

4. Add `Value="50"` to the `<c1:C1ProgressBar>` tag so that the progress bar will appear half-full at run time.
5. Press F5 to run the program.
6. From the drop-down list, select **Vertical**. Observe that the progress bar changes orientations.