

---

ComponentOne

# **Accordion for ASP.NET AJAX**

Copyright © 1987-2010 ComponentOne LLC. All rights reserved.

*Corporate Headquarters*  
**ComponentOne LLC**  
201 South Highland Avenue  
3<sup>rd</sup> Floor  
Pittsburgh, PA 15206 • USA

**Internet:** [info@ComponentOne.com](mailto:info@ComponentOne.com)  
**Web site:** <http://www.componentone.com>

**Sales**

E-mail: [sales@componentone.com](mailto:sales@componentone.com)  
Telephone: 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

**Trademarks**

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

**Warranty**

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

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

**Copying and Distribution**

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

This manual was produced using ComponentOne Doc-To-Help™.

# Table of Contents

<b>ComponentOne Accordion for ASP.NET AJAX</b> .....	<b>1</b>
What's New in Accordion for ASP.NET AJAX.....	1
Revision History.....	1
What's New in 2010 v1.....	1
What's New in 2009 v3.....	1
Installing Studio for ASP.NET AJAX.....	2
Studio for ASP.NET Setup Files.....	2
System Requirements.....	2
Uninstalling Studio for ASP.NET AJAX.....	3
Deploying your Application in a Medium Trust Environment.....	3
End-User License Agreement.....	6
Licensing FAQs.....	6
What is Licensing?.....	6
How does Licensing Work?.....	7
Common Scenarios.....	7
Troubleshooting.....	9
Technical Support.....	11
Redistributable Files.....	11
About This Documentation.....	12
Namespaces.....	12
Creating an AJAX-Enabled ASP.NET Project.....	13
Adding the C1Accordion Component to a Project.....	15
<b>Key Features</b> .....	<b>16</b>
<b>Accordion for ASP.NET AJAX Quick Start</b> .....	<b>21</b>
Step 1 of 5: Adding C1Accordion to the Page.....	21
Step 2 of 5: Customizing the C1Accordion Control.....	21
Step 3 of 5: Working with the Accordion Designer Form.....	23
Step 4 of 5: Adding Content to the C1AccordionPanels.....	24
Step 5 of 5: Running the Project.....	26
<b>C1Accordion Elements</b> .....	<b>29</b>
Header Area.....	29
Content Area.....	30
<b>C1Accordion Design-Time Support</b> .....	<b>30</b>
C1Accordion Smart Tag.....	30
C1Accordion Context Menu.....	32
Accordion Designer Form.....	33
<b>Accordion for ASP.NET AJAX Appearance</b> .....	<b>36</b>
C1Accordion Visual Styles.....	36
Custom Visual Styles.....	38
<b>Expanding and Collapsing C1Accordion</b> .....	<b>39</b>
Expand Direction.....	39
Animation Effects.....	41
Animation Duration.....	43
<b>Client-Side Functionality</b> .....	<b>43</b>
<b>Accordion for ASP.NET AJAX Samples</b> .....	<b>45</b>
<b>Accordion for ASP.NET AJAX Task-Based Help</b> .....	<b>45</b>

Adding Keyboard Accessibility Support to C1Accordion .....	45
Setting the Animation Effects for C1Accordion .....	46
Adding Custom Visual Styles .....	46
Using Built-in Visual Styles .....	48
Changing the C1Accordion Expanding Direction .....	48
Setting a C1AccordionPane's Expand State .....	49
Using Automatic Posts Back to the Server .....	49
Displaying External Content .....	49

# ComponentOne Accordion for ASP.NET AJAX

Display formatted text, images, and Web pages in an organized list of expandable panes using **ComponentOne Accordion for ASP.NET AJAX**. **Accordion for ASP.NET AJAX** provides one control, **C1Accordion**, which allows you to add multiple panes for displaying your information one pane at a time. You can choose from a variety of animation effects, such as bounce or elastic, or customize the look of **C1Accordion** with visual styles to provide a rich interactive user experience. Users simply select a pane to expand it and view the specified information while the other panes are collapsed, automatically organizing your user interface and optimizing the screen real estate.

## Getting Started

To get started, review the following topics:

- [Key Features](#) (page 16)
- [Quick Start](#) (page 21)
- [Samples](#) (page 45)

## What's New in Accordion for ASP.NET AJAX

There were no new features added to **Accordion for ASP.NET AJAX** in v2 2010.



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

## Revision History

The revision history provides recent enhancements to **ComponentOne Accordion for ASP.NET AJAX**.

### What's New in 2010 v1

No new features have been added to **Accordion for ASP.NET AJAX** for the 2010 v1 release of the ComponentOne Studios.

### What's New in 2009 v3

#### Class Members

The following members have been added to **Accordion for ASP.NET AJAX** in the 2009 v3 release:

Member	Description
C1Accordion.OnClientSelectedPaneChanged Property	Handler name of the client <b>SelectedPaneChanged</b> event.

C1Accordion.SelectedPaneChanged Event	Raised when the selected pane changes.
---------------------------------------	--

## Installing Studio for ASP.NET AJAX

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

### Studio for ASP.NET Setup Files

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

- Bin** Contains copies of all binaries (DLLs, Exes) in the ComponentOne Visual Studio ASP.NET package.
- H2Help** Contains online documentation for Studio for ASP.NET 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\\My Documents\ComponentOne Samples

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

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

- Common** Contains support and data files that are used by many of the demo programs.
- Studio for ASP.NET\C1WebUi** Contains a readme.txt file and the folders that make up the Control Explorer and other samples.

Samples can be accessed from the **ComponentOne Sample Explorer**. To view samples, on your desktop, click the **Start** button and then click **All Programs | ComponentOne | Studio for ASP.NET | 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 later  
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 Studio 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 Vista/Windows 7)**.
2. Select **ComponentOne Studio for ASP.NET** 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 existing web\_mediumtrust.config file or create a custom policy file based on the medium trust policy. If you modify the existing web\_mediumtrust.config file, all Web applications will have the same permissions with the permissions you have added. If you want applications to have different permissions, you can instead create a custom policy based on medium trust.

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

In order to add permissions, you can edit the existing web\_mediumtrust.config file. To edit the existing web\_mediumtrust.config file, complete the following steps:

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

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

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

1. Locate the medium trust policy file `web_mediumtrust.config` located by default in the `%windir%\Microsoft.NET\Framework\{Version}\CONFIG` directory.
2. Copy the `web_mediumtrust.config` file and create a new policy file in the same directory.  
Give the new file 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, the ComponentOne licensing model, and frequently asked licensing questions, is available online at <http://www.componentone.com/SuperPages/Licensing/>.

## Licensing FAQs

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

### What is Licensing?

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

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

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

## How does Licensing Work?

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

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

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

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

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

These files are automatically added to the project.

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

The licenses.licx file is a simple text file that contains strong names and version information for each of the licensed components used in the application. Whenever Visual Studio is called upon to rebuild the application resources, this file is read and used as a list of components to query for run-time licenses to be embedded in the appropriate assembly resource. Note that editing or adding an appropriate line to this file can force Visual Studio to add run-time licenses of other controls as well.

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

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

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

## Common Scenarios

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

### *Creating components at design time*

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

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

## Creating components at run time

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

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

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

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

## Inheriting from licensed components

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

- Add a LicenseProvider attribute to the component.

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

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

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

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

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

## Using licensed components in console applications

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

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

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

## Using licensed components in Visual C++ applications

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

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

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

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

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

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

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

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

For example:

```
#if AUTOMATED TESTING
    [AssemblyConfiguration("C1CheckForDesignLicenseAtRuntime")]
#endif
public class MyDerivedControl : C1LicensedControl
{
    // ...
}
```

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

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

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

### **Troubleshooting**

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

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

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

If this happens, there may be a problem with the licenses.licx file in the project. It either doesn't exist, contains wrong information, or is not configured correctly.

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

**If that fails follow these steps:**

1. Open the 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 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 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'll immediately receive a response via e-mail confirming that you've successfully created an incident. This email will provide you with an Issue Reference ID and will provide you with a set of possible answers to your question from our Knowledgebase. You will receive a response from one of the ComponentOne staff members via e-mail in 2 business days or less.
- **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.Controls.4.dll

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

## About This Documentation

### Acknowledgements

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

Firefox is a registered trademark of the Mozilla Foundation.

Safari is a registered trademark of Apple Inc.

### ComponentOne

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

*Corporate Headquarters*

### ComponentOne LLC

201 South Highland Avenue  
3<sup>rd</sup> Floor  
Pittsburgh, PA 15206 • USA  
412.681.4343  
412.681.4384 (Fax)

<http://www.componentone.com>

### ComponentOne Doc-To-Help

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

## Namespaces

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

The general namespace for ComponentOne Web products is **C1.Web**. The following code fragment shows how to declare a **C1Accordion** using the fully qualified name for this class:

- Visual Basic  

```
Dim Accordion As C1.Web.UI.Controls.C1Accordion
```
- C#  

```
C1.Web.UI.Controls.C1Accordion Accordion;
```

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 C1Accordion = C1.Web.UI.Controls.C1Accordion
Imports MyAccordion = MyProject.Objects.C1Accordion

Dim wm1 As C1Accordion
Dim wm2 As MyAccordion
```

- C#

```
using C1Accordion = C1.Web.UI.Controls.C1Accordion;
using MyAccordion = MyProject.Objects.C1Accordion;

C1Accordion wm1;
MyAccordion 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 Accordion 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 controls are 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 Studio 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://www.asp.net/ajax/downloads/archive/>. 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	<a href="#">ASP.NET AJAX Extensions 1.0</a>
Visual Studio 2005	<a href="#">ASP.NET AJAX Extensions 1.0</a> <a href="#">Visual Studio update and add-in</a> (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 **ScriptManager** 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 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 **ScriptManger** 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.

5. 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 **ScriptManger** 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 C1Accordion Component 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, you will notice a **ComponentOne Studio for ASP.NET 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 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 controls to the Toolbox

When you install **ComponentOne Studio for ASP.NET AJAX**, the **C1Accordion** component will appear in the Visual Studio Toolbox customization dialog box.

To manually add the Studio for ASP.NET 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 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, 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.C1Accordion. 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 Controls to the Form

To add Studio for ASP.NET 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** 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
```

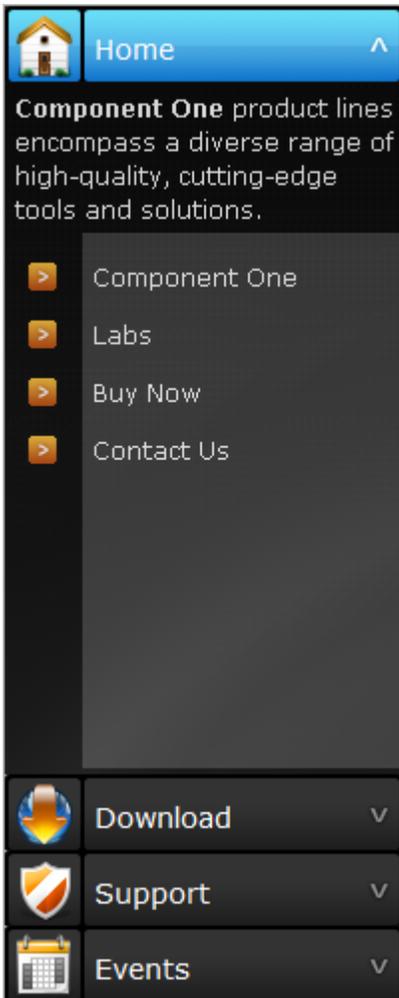
**Note:** This makes the objects defined in the **C1.Web.UI.Controls.2** assembly visible to the project. See [Namespaces](#) (page 12) for more information.

## Key Features

The following are some of the main features of C1Accordion that you may find useful:

- **Display multiple panes at one time**

C1Accordion allows you to provide multiple panes while displaying them at one time.

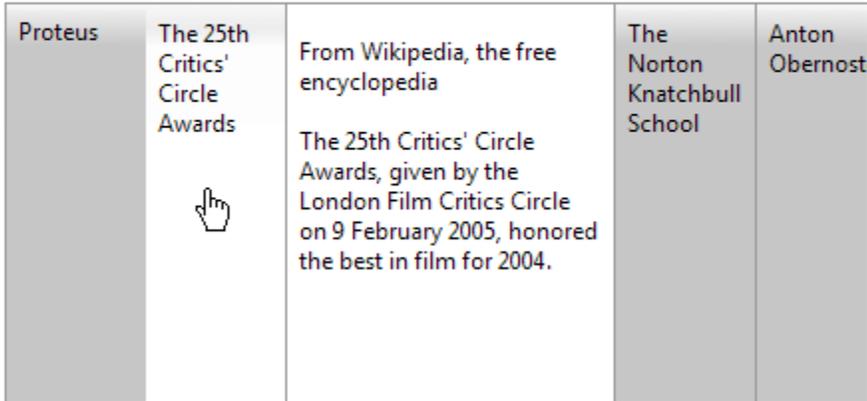


- **Choose from four expanding options for displaying content**

C1Accordion provides four expanding direction options for the **C1AccordionPanes**: **Top**, **Bottom**, **Right**, or **Left**. For example, select **Left** if you want the **C1AccordionPane** to expand so the content area appears on the left.

## C1Accordion - Expand Direction

The C1Accordion supports four expand directions: Top, Right, Bottom, Left.



See [Changing the C1Accordion Expanding Direction](#) (page 48) for more information.

- **Choose from thirty-one built-in animation effects**

C1Accordion includes thirty-one different built-in animations that can be set when the C1AccordionPanels are expanded or collapsed. Easily change animation effects by setting the **ExpandEasing** and **CollapseEasing** properties. For more information, see [Setting the Animation Effects for C1Accordion](#) (page 46).

- **Built-in visual styles**

Choose from five built-in visual styles to quickly change the C1Accordion control's appearance. Visual styles include: **ArcticFox**, **Office2007Black**, **Office2007Blue**, **Office2007Silver**, and **Vista**. See [Using Built-in Visual Styles](#) (page 48) for additional information.

- **Customize the C1Accordion control with visual styles and CSS styling**

Easily customize the C1Accordion control by pointing to your own visual styles using the VisualStyle property.

**Accordion for ASP.NET AJAX** also includes CSS-supported styling so that you can use cascading style sheets to easily style the C1Accordion control to match the design of your current Web site.

See [Adding Custom Visual Styles](#) (page 46) for more information.

- **Keyboard accessibility support**

Add keyboard accessibility support to give the C1Accordion control focus with a specified key combination. At run time, users can use the keyboard arrow keys to navigate through the C1AccordionPanels. For more information, see [Adding Keyboard Accessibility Support to C1Accordion](#) (page 45).

- **Use automatic posts back to the server**

Set the AutoPostBack property to **True** if you want automatic posts back to the server each time a user interacts with the C1Accordion control. See [Using Automatic Posts Back to the Server](#) (page 49) for more information.

- **Display external content**

You can show external content, including content of another Web page in your project or even content of a Web site outside of your project, in the C1AccordionPane content area. Simply set the **ContentUrl** property. See [Displaying External Content](#) (page 49) for more information.



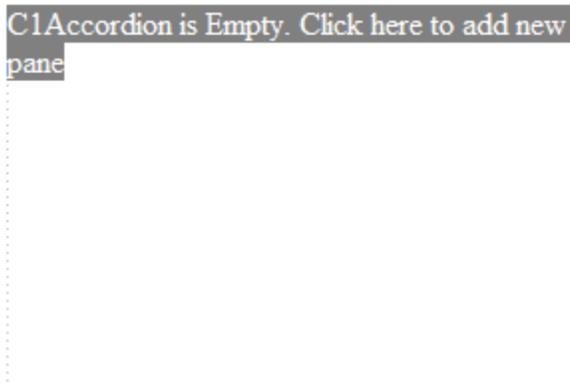
# Accordion for ASP.NET AJAX Quick Start

The C1Accordion Quick Start describes how to get started with the ASP.NET control, **C1Accordion**. In this quick start, you will create an ASP.NET AJAX-Enabled Web Site containing one **C1Accordion** control and four **C1AccordionPanes**. Each accordion pane will allow users to access different areas of a Web site.

## Step 1 of 5: Adding C1Accordion to the Page

In this topic you will add a C1Accordion control to the page.

1. Begin by [creating an ASP.NET AJAX-Enabled Web Site](#) (page 13). Note that if using Visual Studio 2008, you must add a **ScriptManager** control to the form. If using Visual Studio 2005, the **ScriptManager** control is automatically added to the form.
2. While in Design view, navigate to the Visual Studio Toolbox and double-click the **C1Accordion** control to add it to your form. The **C1Accordion** control will be made up of **C1AccordionPanes**.



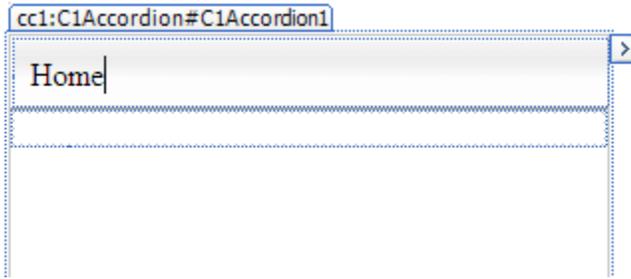
C1Accordion is Empty. Click here to add new pane

**Note:** If the control is not appearing correctly, make sure you have a reference to the C1.Web.UI.Design.2.dll installed with the product in your project.

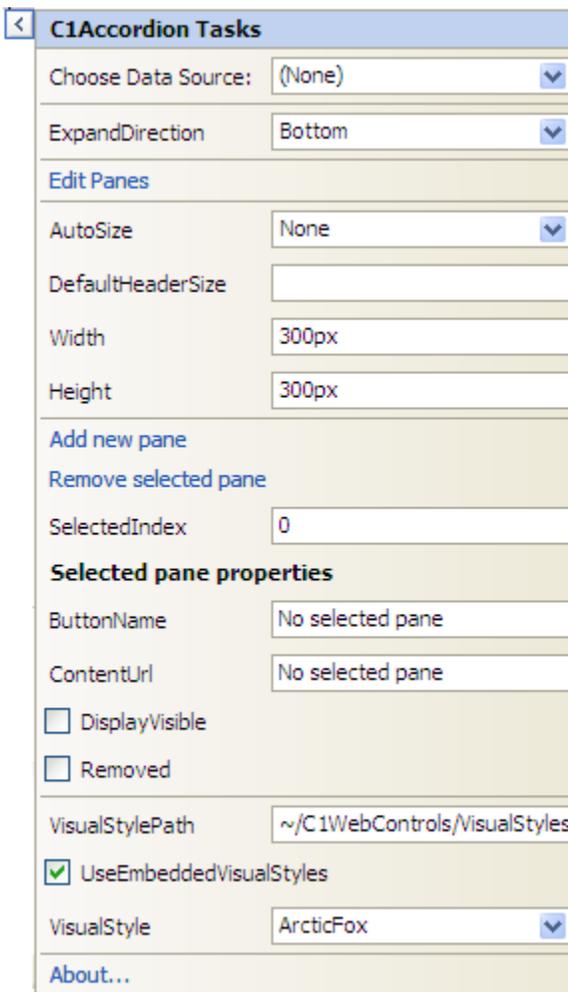
## Step 2 of 5: Customizing the C1Accordion Control

In this topic you will add a **C1AccordionPane** to the C1Accordion control and set some of its properties.

1. Click "**Accordion is Empty. Click here to add new pane**". The **C1Accordion** control now contains a **C1AccordionPane** with a header and content area. The **C1Accordion** control is similar to the **C1NavPanel** control, only it does not contain a navigation pane.
2. Click the header area and enter **Home**.



3. Click the **C1Accordion** smart tag to access the **C1Accordion Tasks** menu where you can set frequently-used properties and tasks.



- a. Set the ExpandDirection property to **Left**. This places the **C1AccordionPanes** vertically, and, when clicked, they expand to the left so the accordion appears to the right of each pane. The accordion can include anything you want, such as text, links, graphics, and so on. You can use HTML and cascading style sheets to format the area as well.
- b. Set the AutoSize property to **Fill** so the size of each pane's content will always be the same regardless of how large or small the horizontal space is for each pane.

- c. Set the `DefaultHeaderSize` property to **50px** so the header text will fit within the panes.
- d. Click the **VisualStyle** drop-down arrow and select **Vista**.
- e. Click **Edit Panes** from the **C1Accordion Tasks** menu. The **Accordion Designer Form** appears.

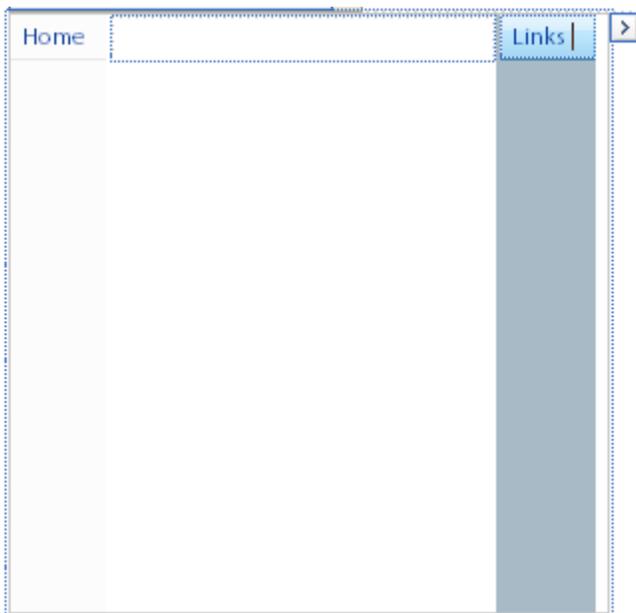
## Step 3 of 5: Working with the Accordion Designer Form

In this topic you will use the **Accordion Designer Form** to add content to a **C1AccordionPane** as well as add an additional **C1AccordionPane** to the **C1Accordion** control.

1. In the **Accordion Designer Form**, select **C1Accordion1Pane1**.
  - a. In the **Misc** category of the designer's Property window, enter <http://www.componentone.com> next to the **ContentUrl** property.

**Note:** Make sure the **Categorized** button is clicked in the designer's Property window.

- b. Right-click **C1Accordion1** and select **Add Child | Pane**.
  - c. Click **OK** to close the designer. The **Home** pane is moved to the left of the **C1Accordion** control, and the new pane appears on the right side of the **C1Accordion** control.
2. Click the second pane and enter **Links**.

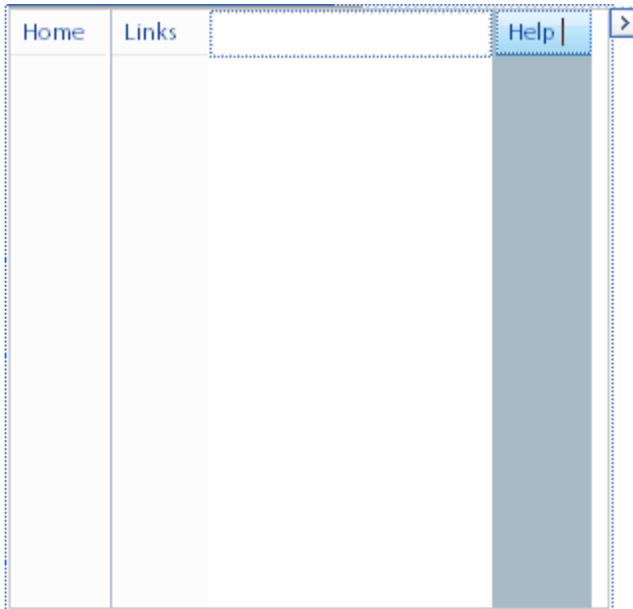


- b. Click the **C1Accordion** smart tag to access the **C1Accordion Tasks** menu again.
  - c. Select **Edit Panes**. The **Accordion Designer Form** appears.
  - d. Set **C1Accordion1Pane2**'s `Expanded` property to **True**.
  - e. Right-click **C1Accordion1** and select **Add Child | Pane** to add another pane.
  - f. Click **OK** to close the designer.

## Step 4 of 5: Adding Content to the C1AccordionPanels

In this topic you will add content to the content area of the **C1AccordionPanels**.

1. Click the new pane and enter **Help**.

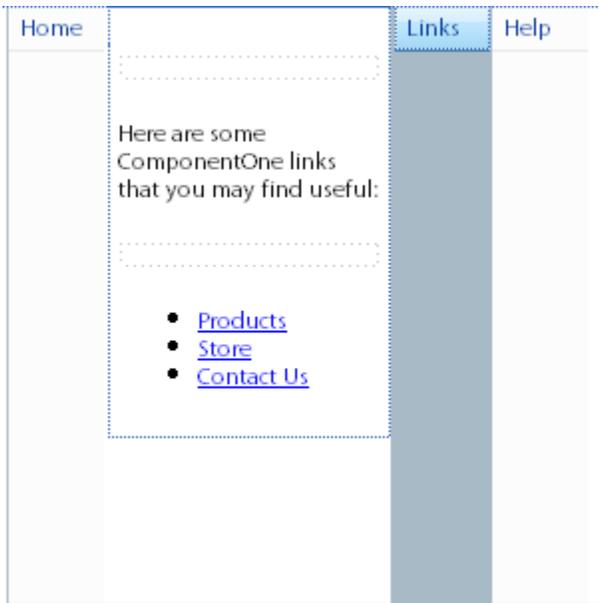


2. To add links in the content area to the left of the **Links** pane, click the **Source** button on your form to switch to Source view, and add some HTML for your links. For example:

```
<cc1:C1AccordionPane ID="C1Accordion1Pane2" runat="server"
Expanded="True">
  <Header>
    Links
  </Header>
  <Content>
    <p></p>Here are some ComponentOne links that you may find
    useful:<p/>
    <ul>
      <li><a
href="http://www.componentone.com/SuperPoducts/StudioEnterprise/">Produ
cts</a></li>
      <li><a
href="http://www.componentone.com/Store.aspx">Store</a></li>
      <li><a
href="http://www.componentone.com/Company/ContactUs.aspx">Contact
Us</a></li>
    </ul>
  </Content>
```

**Note:** Since you have the `ExpandDirection` property set to **Left**, the links are displayed horizontally.

3. Switch to design view and click on the **Links** pane. Notice the content is updated for **C1Accordion1Pane2**.



4. Next we will create an .htm file to add to the Help pane's content.
  - a. In the Visual Studio Solution Explorer, right-click on the name of your project and select **Add New Item**. The **Add New Item** dialog box appears.
  - b. Select **HTML Page**, rename the file **Help.htm** in the **Name** text box, and click **Add**.
  - c. Add the following source code into the Help.htm file and click the **Save** button:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Untitled Page</title>
  <style type="text/css">
    #TextArea1
    {
      height: 111px;
      width: 368px;
    }
    .style3 {
      font-family: Arial, Helvetica, sans-serif;
      font-size: 12px;
    }
  </style>
</head>
<body>
<H2 class="style3">Full-service Support</H2>
<DIV class="style3">ComponentOne services are offered in three
configurations. Our services are offered as Standard, Gold or
a new Platinum level. With these services comes various
combinations of HelpCentral, C1 knowledge base, C1 Forum, on-
line technical support, email technical support, live phone
technical support and so on. In addition to ComponentOne's
services, various combinations of our bug fixes, product
enhancements and on-going minor product upgrades are included.
Finally, with Platinum service and support, ComponentOne also

```

```

provides a 1 year subscription service that provides all
services, all supports, and all next generation product
releases, at a significant discount over the MSRP. </DIV>
<ul>
  <li class="style3">Knowledge Base support</li>
  <li class="style3">ComponentOne Forums provide our customers a
voice</li>
  <li class="style3">HelpCentral is a great resource for
developers</li>
  <li class="style3">Online Web Support</li>
  <li class="style3">Email support provides an easy means of help</li>
  <li class="style3">24 Hour FastTrack Response</li>
  <li class="style3">Industry leading Live Phone Support</li>
</ul>
</body>
</html>

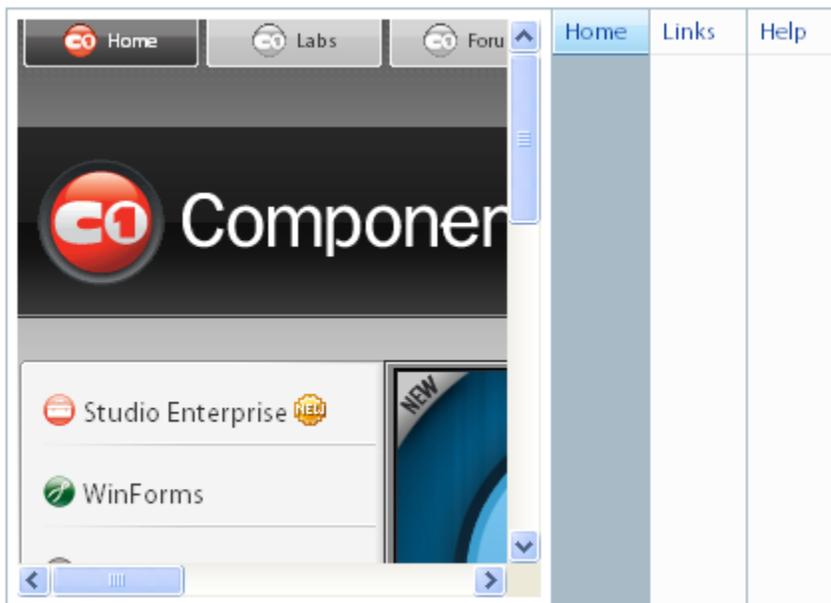
```

5. In the **Default.aspx** page, click the **C1Accordion** smart tag to access the **C1Accordion Tasks** menu again.
6. Select **Edit Panes**. The **Accordion Designer Form** appears.
7. Select the **Help** pane, **C1Accordion1Pane3**, in the treeview.
8. In the **Misc** category, enter **Help.htm** next to the **ContentUrl** property and click **OK** to update the C1Accordion and close the designer.

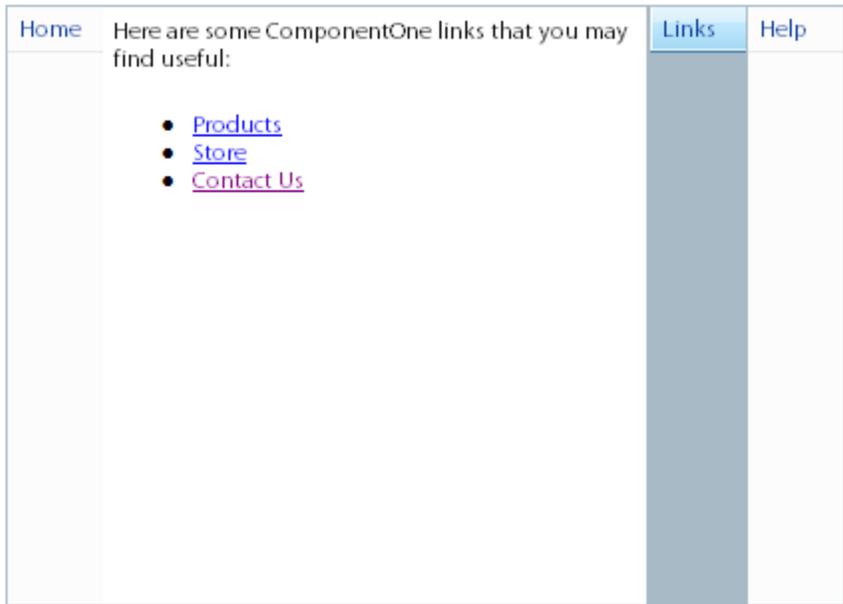
## Step 5 of 5: Running the Project

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

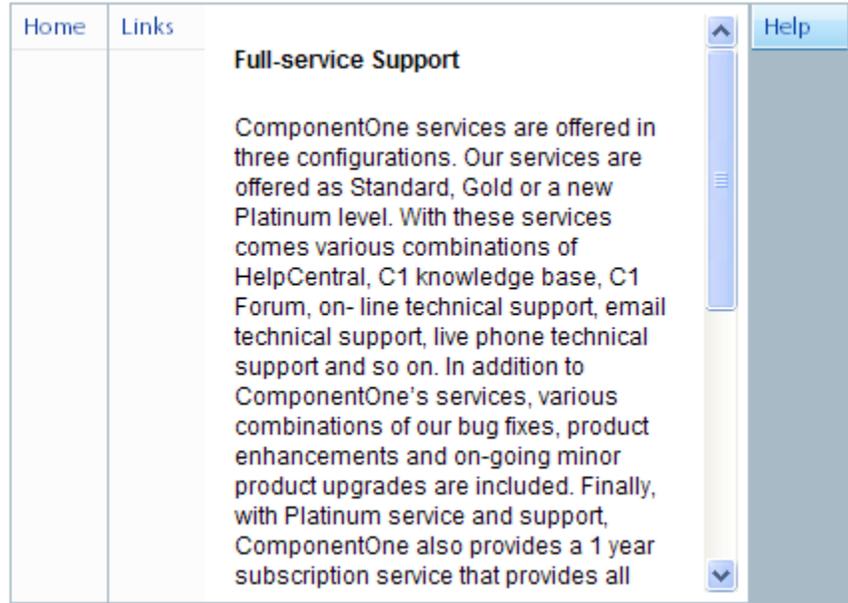
- Click the **Home** pane. The Web site will appear to the left of the pane.



- Click the **Links** pane. The link list will appear to the left of the pane.



- Click the **Help** pane. The text from the Help.htm will appear in the accordion to the left of the pane.



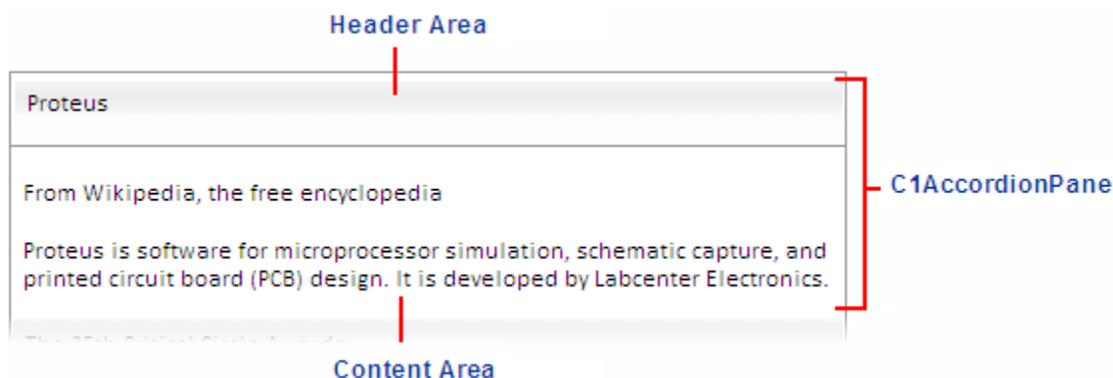
Congratulations! You have successfully completed the **Accordion for ASP.NET AJAX Quick Start**.



# C1Accordion Elements

This section provides a visual and descriptive overview of the elements that comprise the C1Accordion control.

The C1Accordion control is made up of multiple **C1AccordionPanes**. Each **C1AccordionPane** is made up of two distinct elements, the header area and the content area, that represent different aspects of the control.



## Header Area

The **C1AccordionPane's** header area appears at the top of the control and initially displays "C1Accordion is Empty. Click here to add new pane." You can add content, including text, HTML content, images, and arbitrary controls such as buttons and labels, to the header area of C1Accordion. Elements in the header area of the control can be added and moved there through a simple drag-and-drop operation.



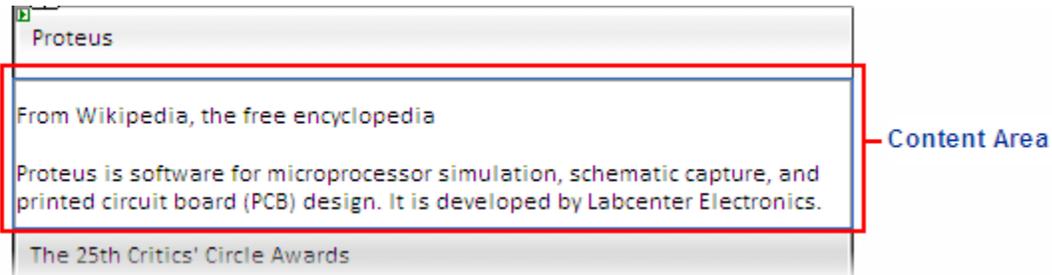
You can add content to the header area of the **C1AccordionPane** at design time. Simply click the header area and begin typing.

When you add content to the header and switch to Source view, you will notice that the header appears inside `<Header>` tags inside the `<ccl:C1AccordionPane>` tags:

```
<ccl:C1AccordionPane runat="server" ID="C1Accordion1Panel1"><Content>
    Content
</Content>
<Header>
    Header
</Header>
</ccl:C1AccordionPane>
```

## Content Area

The **C1AccordionPane**'s content area is initially empty. You can add images, rich text through custom HTML content, URL links through the **ContentUrl** property, and arbitrary controls such as buttons and labels. Elements in the content area of the control can be added and moved there through a simple drag-and-drop operation.



You can add content to the control area of the **C1AccordionPane** at design time. Simply click the content area and begin typing, or add images or other elements as you would normally.

When you add content to the content area and switch to Source view, you will notice that the content appears inside `<Content>` tabs inside the `<cc1:C1AccordionPane>` tags:

```
<cc1:C1AccordionPane runat="server" ID="C1Accordion1Panel"><Content>  
    Content  
</Content>  
<Header>  
    Header  
</Header>  
</cc1:C1AccordionPane>
```

You can also use the **ContentUrl** property to set external content to appear within the content area of the **C1AccordionPane**. For more information, see [Displaying External Content](#) (page 49).

## C1Accordion Design-Time Support

The following sections describe how to use **C1Accordion**'s design-time environment to configure the **C1Accordion** control.

### C1Accordion Smart Tag

The **C1Accordion** 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 **C1Accordion**.

The **C1Accordion** control provides quick and easy access to the **Accordion Designer Form** and common properties through its smart tag.

To access the **C1Accordion Tasks** menu, click the smart tag in the upper-right corner of the **C1Accordion** control.

C1Accordion Tasks	
Choose Data Source:	(None) ▼
ExpandDirection	Bottom ▼
<b>Edit Panes</b>	
AutoSize	None ▼
DefaultHeaderSize	
Width	300px
Height	300px
<b>Add new pane</b>	
<b>Remove selected pane</b>	
SelectedIndex	0
<b>Selected pane properties</b>	
ButtonName	No selected pane
ContentUrl	No selected pane
<input type="checkbox"/> DisplayVisible	
<input type="checkbox"/> Removed	
VisualStylePath	~/C1WebControls/VisualStyles
<input checked="" type="checkbox"/> UseEmbeddedVisualStyles	
VisualStyle	ArcticFox ▼
<a href="#">About...</a>	

The **C1Accordion Tasks** menu operates as follows:

- **Choose Data Source**

Clicking the drop-down arrow in the **Choose Data Source** box opens a list of available data sources and allows you to add a new data source. To add a new data source to the project, click **<New data source>** to open the **Data Source Configuration Wizard**.

- **ExpandDirection**

Set this property to determine where the accordion will appear when a **C1AccordionPane** is clicked. Options include: **Top**, **Right**, **Bottom**, and **Left**.

- **Edit Panes**

Clicking **Edit Panes** opens the **Accordion Designer Form**, where panes can be added, removed, and reordered. You can preview the **C1Accordion** here as well as set a variety of properties defining the appearance, behavior, and more for each **C1AccordionPane**.

- **AutoSize**

The **AutoSize** property determines the width and height of the accordion of the **C1AccordionPanes**. If set to **None** (default), then the accordion will increase as necessary to fit the pane's contents. If set to **Fill**, then the accordion size will always be equal to its height and width for horizontal orientation.

- **DefaultHeaderSize**  
This property determines the size of the **C1AccordionPane**.
- **Width**  
Set this property to determine the width of the C1Accordion control.
- **Height**  
Set this property to determine the height of the C1Accordion control.
- **Add new pane**  
Clicking **Add new pane** simply adds an additional **C1AccordionPane** to the C1Accordion control.
- **Remove selected pane**  
Clicking **Remove selected pane** deletes the currently selected **C1AccordionPane** in the C1Accordion control.
- **SelectedIndex**  
Set this property to determine the **C1AccordionPane** to be initially displayed.
- **ButtonName**  
Set this property to determine the name for the selected **C1AccordionPane**.
- **ContentUrl**  
Set this property to the URL of the Web site you wish to display or provide your own .htm file to display formatted content, links, images, and so on.
- **DisplayVisible**  
Unchecking the **DisplayVisible** checkbox hides the currently selected **C1AccordionPane**.
- **Removed**  
Hides the current **C1AccordionPane**.
- **VisualStylePath**  
The VisualStylePath property specifies the location of the visual styles used for the control. The default value for VisualStylePath property is ~/C1WebControls/VisualStyles. If you create a custom style, add it to this location ~/VisualStyles/StyleName/C1Accordion/styles.css, set the **VisualStylePath** property to ~/VisualStyles, and set the **VisualStyle** property to **StyleName** (assuming that **StyleName** is the name used to define the style in the style.css file). Uncheck the **UseEmbeddedVisualStyles** property.
- **UseEmbeddedVisualStyles**  
This checkbox is checked by default so that the visual styles embedded in C1.Web.UI.Controls.2.dll, such as **ArcticFox** and **Vista**, can be used. If you want to use your own custom styles, uncheck this checkbox and specify the location of your visual styles using the VisualStylePath property.
- **VisualStyle**  
Clicking the **VisualStyle** drop-down arrow allows you to select from various built-in visual styles. See [Using Built-in Visual Styles](#) (page 48) for more information.

## C1Accordion Context Menu

C1Accordion has additional commands available on the context menu that Visual Studio provides for all .NET and ASP.NET controls.

Right-click anywhere on the C1Accordion control to display the **C1Accordion** context menu. The context menu commands operate as follows:

- **Edit Panes**

Clicking **Edit Panes** opens the **Accordion Designer Form**, allowing you to access the properties for the **C1AccordionPanes**.

- **Add new pane**

Clicking **Add new pane** simply adds an additional **C1AccordionPane** to the **C1Accordion** control.

- **Remove selected pane**

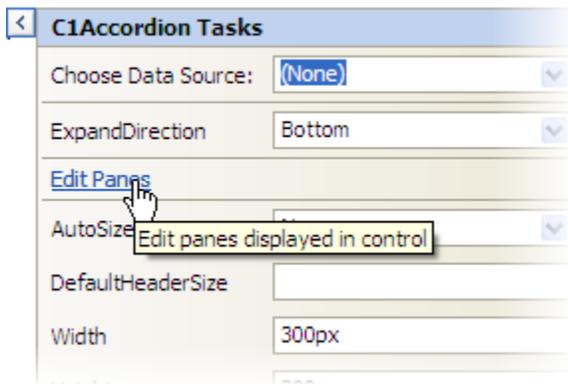
Clicking **Remove selected pane** deletes the currently selected **C1AccordionPane** in the **C1Accordion** control.

## Accordion Designer Form

The **Accordion Designer Form** is **C1Accordion's** designer for editing its properties, as well as the **C1AccordionPane** properties. The **Accordion Designer Form** is similar to the Properties window as it allows programmers to modify the control visually. However, it allows you to select a **C1AccordionPane**, set its properties, manipulate the panes, and then preview the appearance of the **C1Accordion** control, all within the form.

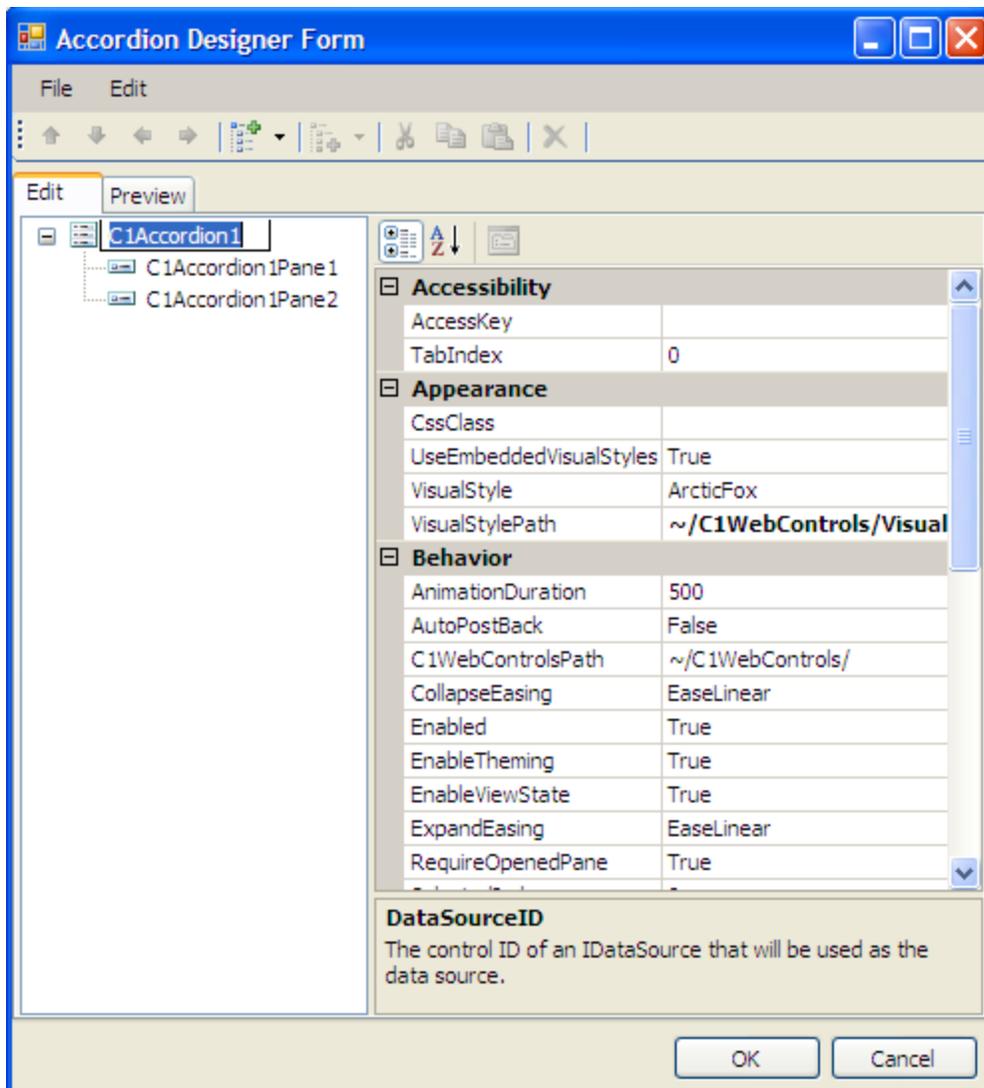
In this topic you will become familiar with the **Accordion Designer Form's** design interface so you can use the commands within it to edit **C1Accordion** with minimal effort and time.

To open the **Accordion Designer Form**, click the **C1Accordion** smart tag and select the **Edit Panes** link from the **C1Accordion Tasks** menu:



### Exploring the Accordion Designer Form

The **Accordion Designer Form** contains a menu, toolbar, **Edit** tab, **Preview** tab, and properties pane.



- **Accordion Designer Form Menu**

The **Accordion Designer Form** menu contains the following menu items and subitems:

Menu Item	Submenu Item	Description
File	Load from XML	Load the formatting for a C1Accordion control from an .xml file.
	Save as XML	Save the current formatting of the C1Accordion control to an .xml file.
	Exit	Closes the <b>Accordion Designer Form</b> .
Edit	Insert Item	Inserts a new <b>C1AccordionPane</b> at the specified place in the list of panes.
	Add Child	Adds a new <b>C1Accordion</b> pane as a child of the C1Accordion.
	Cut	Cuts the selected <b>C1AccordionPane</b> to be moved in the list of panes.

	Copy	Copies the selected <b>C1AccordionPane</b> .
	Paste	Pastes a <b>C1AccordionPane</b> at the specified location in the list of panes.
	Delete	Removes the selected <b>C1AccordionPane</b> .
	Rename	Allows you to change the name of the <b>C1AccordionPane</b> .

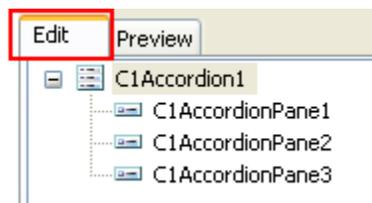
- **Accordion Designer Form Toolbar**



The table below describes each button in the toolbar:

Button	Name	Description
	Move Item Up	Moves the selected <b>C1AccordionPane</b> up the list of panes.
	Move Item Down	Moves the selected <b>C1AccordionPane</b> down the list panes.
	Move Item Left	Moves the selected <b>C1AccordionPane</b> to the left in the hierarchy.
	Move Item Right	Moves the selected <b>C1AccordionPane</b> to the right in the hierarchy.
	Add Child Item	Inserts a <b>C1AccordionPane</b> as a child of the C1Accordion control.
	Insert Item	Inserts a <b>C1AccordionPane</b> at the specified location in the list of panes.
	Cut	Cuts the selected <b>C1AccordionPane</b> to be moved in the list of panes.
	Copy	Copies the selected <b>C1AccordionPane</b> .
	Paste	Pastes a <b>C1AccordionPane</b> at the specified location in the list of panes.
	Delete	Removes the selected <b>C1AccordionPane</b> .

- **Edit Tab**



Click the **Edit** tab and select the C1Accordion control or the desired **C1AccordionPane** for which you would like to manipulate or adjust the properties.

- **Preview Tab**

Click the **Preview** tab for a preview of what the C1Accordion control will look like.

- **Properties Pane**

The **Accordion Designer Form** properties pane is almost identical to the Visual Studio Properties window. Simply select a **C1AccordionPane** or the **C1Accordion** control and set the desired properties here.

- **Command Buttons**

The command buttons are summarized in the following table:

Button	Description
<b>OK</b>	Clicking <b>OK</b> applies the new settings to the <b>C1Accordion</b> control.
<b>Cancel</b>	Clicking <b>Cancel</b> closes the <b>Accordion Designer Form</b> , cancelling the new settings and applying the default settings to the <b>C1Accordion</b> control.

# Accordion for ASP.NET AJAX

## Appearance

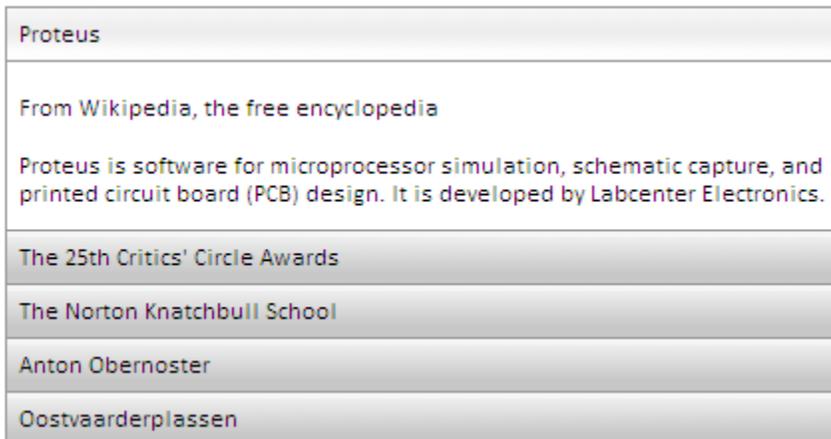
The following sections describe how to change the appearance of **C1Accordion** using built-in visual styles as well as how to use your own custom styles.

### C1Accordion Visual Styles

**ComponentOne Accordion for ASP.NET AJAX** provides five built-in visual styles, allowing you to automatically format the **C1Accordion** control. The styles include the following: **ArcticFox**, **Office2007Black**, **Office2007Blue**, **Office2007Silver**, and **Vista**. For more information, see [Using Built-in Visual Styles](#) (page 48).

#### ArcticFox Style

The following image displays the **ArcticFox** style. This is the default format of the **C1Accordion** control:



#### Office2007Black Style

The following image displays the **Office2007Black** style:

<b>Proteus</b>
From Wikipedia, the free encyclopedia
Proteus is software for microprocessor simulation, schematic capture, and printed circuit board (PCB) design. It is developed by Labcenter Electronics.
The 25th Critics' Circle Awards
The Norton Knatchbull School
Anton Obernoster
Oostvaarderpassen

### Office2007Blue Style

The following image displays the **Office2007Blue** style:

<b>Proteus</b>
From Wikipedia, the free encyclopedia
Proteus is software for microprocessor simulation, schematic capture, and printed circuit board (PCB) design. It is developed by Labcenter Electronics.
The 25th Critics' Circle Awards
The Norton Knatchbull School
Anton Obernoster
Oostvaarderpassen

### Office2007Silver Style

The following image displays the **Office2007Silver** style:

<b>Proteus</b>
From Wikipedia, the free encyclopedia
Proteus is software for microprocessor simulation, schematic capture, and printed circuit board (PCB) design. It is developed by Labcenter Electronics.
The 25th Critics' Circle Awards
The Norton Knatchbull School
Anton Obernoster
Oostvaarderpassen

## Vista Style

The following image displays the **Vista** style:



**Note:** Additional styles are installed with the product, but they must be added as custom visual styles at this time. These styles include: **BureauBlack**, **Evergreen**, **ExpressionDark**, **RainierOrange**, **RainierPurple**, **ShinyBlue**, and **Windows7**. See [Adding Custom Visual Styles](#) (page 46) for more information.

## Custom Visual Styles

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

 **Tip:** The easiest way to create a custom visual styles is by modifying one of the control's pre-existing visual styles. You can find the .css sheets and images for the Accordion for ASP.NET visual styles within the installation directory at C:\Program Files\ComponentOne\Studio for ASP.NET\C1WebUI\VisualStyles. If you use one of the existing .css sheets, you must change any instances of the default file name, for example **\_ArcticFox**, to your new style name, for example **\_CustomStyle**, in order for your custom style to work.

Before adding your .css file and images, you will have to create a hierarchy of folders, the last of which will hold 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 "C1Accordion". The image folder and .css file should be placed underneath the **C1Accordion** folder. The result will resemble the following:



This structure of these folders is very important; **Accordion for ASP.NET AJAX** will always look for the `~/VisualStyles/StyleName/C1Accordion/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.

For more information on customizing the appearance of **Accordion for ASP.NET AJAX** controls, see [Adding Custom Visual Styles](#) (page 46).

# Expanding and Collapsing C1Accordion

The following topics describe the direction options for expanding the C1Accordion control, the different animation effects, and the duration of the animation effects.

## Expand Direction

C1Accordion provides four options for expanding a **C1AccordionPane**: **Top**, **Bottom**, **Right**, or **Left**. The [Changing the C1Accordion Expanding Direction](#) (page 48) topic shows you how to set the **ExpandDirection** property.



*Expand Top*

## C1Accordion - Expand Direction

The C1Accordion supports four expand directions: Top, I

Proteus
From Wikipedia, the free encyclopedia  The 25th Critics' Circle Awards, given by the London Film Critics Circle on 9 February 2005, honored the best in film for 2004.
The 25th Critics' Circle Awards 
The Norton Knatchbull School

*Expand Bottom*

## C1Accordion - Expand Direction

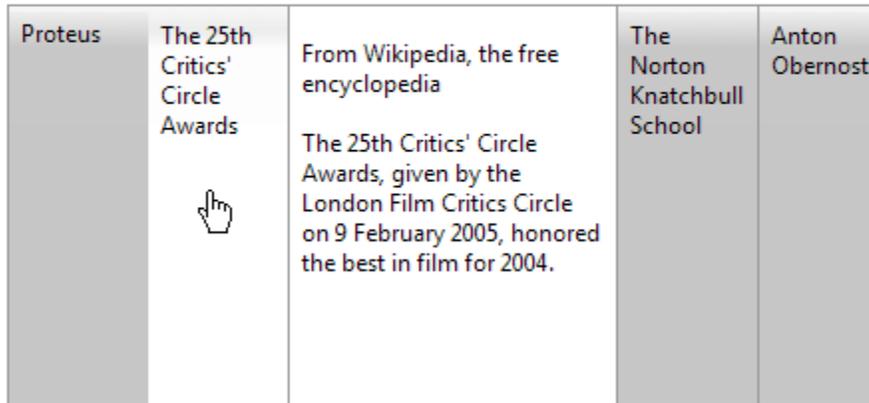
The C1Accordion supports four expand directions: Top, Right, Bottom. Left.

Proteus	From Wikipedia, the free encyclopedia  The 25th Critics' Circle Awards, given by the London Film Critics Circle on 9 February 2005, honored the best in film for 2004.	The 25th Critics' Circle Awards 	The Norton Knatchbull School	Anton Obernc
---------	--	--	------------------------------	--------------

*Expand Right*

## C1Accordion - Expand Direction

The C1Accordion supports four expand directions: Top, Right, Bottom, Left.



*Expand Left*

## Animation Effects

C1Accordion includes thirty-one different built-in animation options that allow you to customize how animation effects are transitioned in the C1Accordion control. You can change how the control expands and collapses by setting the `ExpandEasing` and `CollapseEasing` properties. By default, the `ExpandEasing` and `CollapseEasing` properties are set to `EaseLinear`, and the control expands and collapses with a smooth linear transition effect.

The [Setting the Animation Effects for C1Accordion](#) (page 46) topic shows you how to set the `ExpandEasing` property.



### Sample Project Available

For a demonstration of each transition effect possible in the C1Accordion control, see the **C1Accordion Animation** page of the **ControlExplorer** sample.

The following table describes each transition effect choice:

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 Duration

You can set how long the C1Accordion expand and collapse animation effects take by using the AnimationDuration property. The unit of time used for specifying animation effect duration is in milliseconds, and the default setting is 500 milliseconds, or half of a second. Increase this value for a longer animation effect, and decrease this number for a shorter animation effect.

The [Setting the Animation Effects for C1Accordion](#) (page 46) topic shows you how to set the AnimationDuration property.



### Sample Project Available

For a demonstration of setting the AnimationDuration property, see the **C1Accordion Animation** page of the **ControlExplorer** sample.

# Client-Side Functionality

C1Accordion includes a robust client-side object model, where a majority of server-side properties can be set on the client side.

When a C1Accordion control is added to a Web project, an instance of the client-side control will be created automatically. You can then access properties and methods of the C1Accordion control without having to postback to the server, increasing the efficiency of your Web site and providing a better user experience.

Suppose a C1Accordion control with the name **C1Accordion1** is added to a Web page. When the page is rendered, a corresponding **C1Accordion** object will be created. Use the following syntax to get the client object:

```
$get("C1Accordion1").control
```

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).

For more information about the client-side properties and methods, see the C1Accordion client-side reference section of this documentation.



# Accordion 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.

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

The following pages within the ControlExplorer sample installed with **ComponentOne Studio for ASP.NET AJAX** detail the C1Accordion control's functionality:

## C# Sample

Sample	Description
Animation	Shows the different animation effects for the C1Accordion control.
ExpandDirection	Shows the C1Accordion expanding options: Top, Right, Bottom, and Left.
VisualStyles	Displays the built-in C1Accordion control Visual Styles including: Arctic Fox, Office2007Black, Office2007Blue, Office2007Silver, and Vista.

# Accordion for ASP.NET AJAX Task-Based Help

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

Each task-based help topic also assumes that you have created a new AJAX-enabled ASP.NET project. **ComponentOne Accordion 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 C1Accordion 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. For additional information on this topic, see [Creating an AJAX-Enabled ASP.NET Project](#) (page 13).

## Adding Keyboard Accessibility Support to C1Accordion

You can make C1Accordion and each of its **C1AccordionPanels** accessible through the keyboard by setting the **AccessKey** property.

### To set the access key using the Accordion Designer Form:

1. Click the C1Accordion smart tag to open the **C1Accordion Tasks** menu.

2. Select **Edit Panes** and choose **C1Accordion1** in the **Accordion Designer Form**. Note that you can also add keyboard access to each of the **C1AccordionPanes**.
3. Enter the desired letter next to the `AccessKey` property.
4. Click **OK** to close the **Accordion Designer Form**. When you run the project and press the ALT key plus the letter key entered in the previous step, **C1Accordion1** gets the focus.

#### To set the access key programmatically:

Add the following code to your form:

- Visual Basic

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
    C1Accordion1.AccessKey = "M"
End Sub
```

- C#

```
protected void Page_Load(object sender, EventArgs e)
{
    C1Accordion1.AccessKey = "M";
}
```

## Setting the Animation Effects for C1Accordion

You can add animation effects to the `C1Accordion` control by setting any of the animation properties: `CollapseEasing`, `ExpandEasing`, and `AnimationDuration`.

1. Click the `C1Accordion` smart tag to open the **C1Accordion Tasks** menu.
2. Select **Edit Panes** and choose **C1AccordionPanel1**.
3. Set the `AnimationDuration` property to "1500". This will make the duration of the **C1AccordionPane** collapse a little slower so you can see the animation effect.
4. Set the `ExpandEasing` property to **EaseInBounce**, for example, or one of the other options.
5. Click **OK** to close the **Accordion Form Designer**. When you run your project and click the first **C1AccordionPane**, notice how it bounces when expanded.

## Adding Custom Visual Styles

You can use the `VisualStyle`, `VisualStylePath`, and `UseEmbeddedVisualStyles` properties to create a custom visual style for your `C1Accordion`. This topic assumes you have already added a `C1Accordion` control to your page. For more information on custom visual styles, see [Custom Visual Styles](#) (page 38).

To add a custom visual style, best practice is to copy one of the existing visual styles and customize it. In this example we will use the **C1Accordion ArcticFox** style.

#### Adding Custom Visual Styles in Design View

To add a custom visual style, complete the following steps:

1. Copy the theme folder `C:\Program Files\ComponentOne\Studio for ASP.NET\C1WebUI\VisualStyles\ArcticFox\C1Accordion` to a new folder in your Visual Studio project so the folder structure is: `~\VisualStyles\CustomStyle\C1Accordion`.



2. Open the styles.css file in the ~/VisualStyles/CustomStyle/C1Accordion folder and replace any instance of "ArcticFox" with "CustomStyle". You can modify the CSS definition to customize the appearance.
3. Locate the first instance of **.C1Accordion\_CustomStyle** and change the border attribute so it looks like the following:

```

/* ===== CustomStyle C1Accordion
===== */
.C1Accordion_CustomStyle
{
    border: solid 5px #900;
}

```

4. Save and close the styles.css file.
5. Select the C1Accordion control on your form and click the smart tag to open the **C1Accordion Tasks** menu.
  - a. Uncheck the **UseEmbeddedVisualStyles** property to set it to **False**.
  - b. Make sure the **VisualStylePath** property is set to ~/VisualStyles.
  - c. Select **CustomVisualStyle (external)** from the **VisualStyle** property drop-down list.
6. Press F5 to run your project and notice the C1Accordion control has a red border.

#### Adding Custom Visual Styles in Source View

To add a custom visual style in Source view, complete the following steps:

1. Complete steps 1 through 4 under **Adding Custom Visual Styles in Design View**.
2. Click the **Source** tab to open the Source source view and enter `VisualStyle="CustomStyle"`, `VisualStylePath="~/VisualStyles"`, and `UseEmbeddedVisualStyles="False"` into the `<c1:C1Accordion>` tag. Your XHTML will resemble the following:

```

<c1:C1Accordion ID="C1Accordion1" runat="server" Height="22px"
VisualStyle="CustomStyle" VisualStylePath="~/VisualStyles"
UseEmbeddedVisualStyles="False" Width="155px" />

```

#### Adding Custom Visual Styles in Code

To add a custom visual style, complete the following steps:

1. Complete steps 1 through 4 under **Adding Custom Visual Styles in Design View**.
2. Double-click the Web project to place a **Page\_Load** event in the code editor.
3. Set the **UseEmbeddedVisualStyles** property to **False** by adding the following code to the **Page\_Load** event:

- Visual Basic

```
C1Accordion1.UseEmbeddedVisualStyles = False
```

- C#

```
C1Accordion1.UseEmbeddedVisualStyles = false;
```

4. Change the **VisualStylePath** property:

- Visual Basic

```
C1Accordion1VisualStylePath = "~/VisualStyles"
```

- C#

```
C1Accordion1.VisualStylePath = "~/VisualStyles";
```

5. Select the custom visual style:

- Visual Basic

```
C1Accordion1.VisualStyle = "CustomStyle"
```

- C#

```
C1Accordion1.VisualStyle = "CustomStyle";
```

6. Run the program and observe that the C1Accordion control has adopted your custom visual style.

## Using Built-in Visual Styles

You can format C1Accordion with one of five built-in visual styles. Note that you can also use your own style; see [Adding Custom Visual Styles](#) (page 46) for more information.

### Changing the Visual Style Using the Smart Tag

You can change the style of C1Accordion at design time using the **C1Accordion Tasks** menu.

1. Click the C1Accordion smart tag to open the **C1Accordion Tasks** menu.
2. Click drop-down arrow next to **VisualStyle**.
3. Select one of the built-in styles listed. The style is applied to the C1Accordion control.

### Changing the Visual Style Programmatically

To change the style of C1Accordion programmatically, use the following code. In this example, "Vista" is used, but you can replace it with any of the built-in styles (**ArcticFox**, **Office2007Black**, **Office2007Blue**, **Office2007Silver**, or **Vista**) or use your own style.

- Visual Basic

```
C1Accordion1.VisualStyle = "Vista"
```

- C#

```
C1Accordion1.VisualStyle = "Vista";
```

## Changing the C1Accordion Expanding Direction

You can easily change the direction in which the content area of a **C1AccordionPane** expands. Simply set the **ExpandDirection** property.

### To change the expand direction using the C1Accordion Tasks menu:

1. Click the C1Accordion smart tag to open the **C1Accordion Tasks** menu.
2. Click the drop-down arrow next to the **ExpandDirection** property and select one of the options.

### To change the expand direction programmatically:

Add the following code to your form:

- Visual Basic

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As  
System.EventArgs) Handles Me.Load  
C1Accordion1.ExpandDirection =  
C1.Web.UI.Controls.C1Expander.C1ExpandDirection.Left  
End Sub
```

- C#

```
protected void Page_Load(object sender, EventArgs e)
{
    C1Accordion1.ExpandDirection =
    C1.Web.UI.Controls.C1Expander.C1ExpandDirection.Left;
}
```

## Setting a C1AccordionPane's Expand State

You can determine which **C1AccordionPane** you would like to have expanded when a page is loaded using the **Expanded** property. By default, the **Expanded** property is set to **True** for **C1Accordion1Pane1**. If you would like a different **C1AccordionPane** expanded when the page loads, complete the following steps:

1. Click the C1Accordion smart tag to open the **C1Accordion Tasks** menu.
2. Select **Edit Panes** and choose a **C1AccordionPane**.
3. Set the **Expanded** property to **True**.
4. Set the **Expanded** properties for the other panes to **False**.

## Using Automatic Posts Back to the Server

You can have C1Accordion perform automatic posts back to the server each time a user interacts with the C1Accordion control. To do this, set the **AutoPostBack** property.

**To set the AutoPostBack property using the C1Accordion Tasks menu:**

1. Click the C1Accordion smart tag to open the **C1Accordion Tasks** menu.
2. Select **Edit Panes**. The **Accordion Designer Form** appears.
3. Select **C1Accordion1** on the **Edit** tab.
4. Set the **AutoPostBack** property to **True** in the properties pane. If the properties are categorized, you will find it under the **Behavior** category.

**To set the AutoPostBack property programmatically:**

Add the following code to your form:

- Visual Basic

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
    C1Accordion1.AutoPostBack = True
End Sub
```

- C#

```
protected void Page_Load(object sender, EventArgs e)
{
    C1Accordion1.AutoPostBack = "True";
}
```

## Displaying External Content

You can show external content, including content of another Web page in your project or even content of a Web site outside of your project, in the C1AccordionPane content area.

**To display external content in a C1AccordionPane's accordion using the C1Accordion Tasks menu:**

1. Click the C1Accordion smart tag to open the **C1Accordion Tasks** menu.

2. Enter a URL next to the **C1AccordionPane's ContentUrl** property.

**To display external content in a C1AccordionPane's accordion programmatically:**

Add the following code to your form:

- Visual Basic

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As  
System.EventArgs) Handles Me.Load  
    C1AccordionPanel1.ContentUrl = "http://www.componentone.com"  
End Sub
```

- C#

```
protected void Page_Load(object sender, EventArgs e)  
{  
    C1AccordionPanel1.ContentUrl = "http://www.componentone.com";  
}
```