ComponentOne

SuperPanel for ASP.NET

Corporate Headquarters
ComponentOne LLC
201 South Highland Avenue
3rd Floor
Pittsburgh, PA 15206 • USA

Internet: info@ComponentOne.com
Web site: http://www.componentone.com

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

Trademarks

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

Warranty

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

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

Copying and Distribution

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

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

Table of Contents

ComponentOne SuperPanel for ASP.NET Overview 1
- What's New in ComponentOne SuperPanel for ASP.NET 1
- Installing SuperPanel for ASP.NET 2
- SuperPanel for ASP.NET Setup Files 2
- System Requirements 3
- Uninstalling Super Panel for ASP.NET 3
- Deploying your Application in a Medium Trust Environment 4
- End-User License Agreement 7
- Licensing FAQs 7
- What is Licensing? 7
- How does Licensing Work? 7
- Common Scenarios 8
- Troubleshooting 10
- Technical Support 11
- Redistributable Files 12
- About This Documentation 12
- Namespaces 13
- Creating an AJAX-Enabled ASP.NET Project 14
- Adding Super Panel for ASP.NET Component to a Project 16

Key Features 17

SuperPanel for ASP.NET Quick Start 20
- Step 1 of 4: Adding C1SuperPanel to the Page 20
- Step 2 of 4: Adding Content to the C1SuperPanel Control 21
- Step 3 of 4: Creating a Mixed Scroll Mode in Code 22
- Step 4 of 4: Adding Animation Effects to C1SuperPanel 24

SuperPanel for ASP.NET Top Tips 26

C1SuperPanel Elements 27
- Content Element 27
- Custom ScrollBar Element 28

Design-Time Support 28
- C1SuperPanel Smart Tag 28

Super Panel for ASP.NET Appearance 29
- Visual Styles 29
- Custom Visual Styles 30
- Content Template 31
- Rounded Corners 31
- Drop Shadow 32

SuperPanel Behavior 32
- SuperPanel Sizing 32
- SuperPanel Custom ScrollBar 32
- SuperPanel Scroll Settings 33
- Scrolling Elements 34
- Scrolling Animation 35
- Scrolling Animation Transitions 35
- Scrolling Animation Duration 36

Client-Side Functionality 37

Client-Side Properties 37
Using the C1SuperPanelExtender Control 38
SuperPanel for ASP.NET Samples 40
SuperPanel for ASP.NET Task-Based Help 40
Creating a C1SuperPanel in Code 41
Disabling Vertical Scrolling 42
Wrapping an ASP.NET Control Using C1SuperPanelExtender 42
Customizing C1SuperPanel's Appearance 43
Adding Custom Visual Styles 43
Applying the Visual Style to Children Elements in the SuperPanel 44
Setting the Visual Style 46
Changing the Font 47
Creating a Content Template 48
Setting the Content Background Color 49
Setting C1SuperPanel Behaviors 50
Using the System ScrollBars on SuperPanel 50
Adding Keyboard Support 50
Animate Scrolling Within an Element 50
Client-Side Tasks 52
Set the Height and Width at Run Time 52
Specify Target Scroll Position 53

ComponentOne SuperPanel for ASP.NET Overview

ComponentOne SuperPanel for ASP.NET enables you to easily scroll overflowed elements. It provides you with different options to customize the scrollbars and scrolling effects.

ComponentOne SuperPanel for ASP.NET includes one control, **C1SuperPanel**, which includes several built-in animation and styling effects to easily customize your application. An extender control is included to attach to the C1SuperPanel control. The extender control can be used to wrap any ASP.NET control. With CSS styling and built-in Office 2007 and Vista themes, you can easily create sophisticated scrolling effects.

Getting Started

Get started with the following topics:

- Key Features (page 17)
- SuperPanel for ASP.NET Quick Start (page 20)
- SuperPanel for ASP.NET Samples (page 40)

What's New in ComponentOne SuperPanel for ASP.NET

This documentation was last revised on February 26, 2010 for the 2010 v1 release. A new feature, class member, and task-based help have been added to C1SuperPanel in this release.

New Features

The following new feature was added to **ComponentOne SuperPanel for ASP.NET** in the 2010 v1 release:

- You can now apply the visual style to children elements in the SuperPanel control using the ApplyVisualStyleToChildren property.

  The following image shows the ApplyVisualStyleToChildren property used to apply C1SuperPanel's visual style to its C1 children controls such as C1Menu:

To see how to use the ApplyVisualStyleToChildren property see Applying the Visual Style to Children Elements in the SuperPanel (page 44).

New Members

The following member was added to **ComponentOne SuperPanel for ASP.NET**:

Member	Description
ApplyVisualStyleToChildren	Gets or sets a value indicating whether to apply a visual style to children C1Controls.

New Task-Based Help Topics

The following task-based help topics were added to **ComponentOne SuperPanel for ASP.NET**:

Task-Based Help Topic	Description
Applying the Visual Style to Children Elements in the SuperPanel (page 44)	Shows how to use the ApplyVisualStyleToChildren property to apply C1SuperPanel's visual style to C1 children controls.
Disabling Vertical Scrolling (page 42)	Shows how to use the **VScroller.Visibility** property to disable the vertical scrolling on the C1SuperPanel control.
Wrapping an ASP.NET Control Using C1SuperPanelExtender (page 42)	Shows how to wrap an ASP.NET control such as a Label control in the C1SuperPanel control using the **TargetControlID** property.

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.

Installing SuperPanel for ASP.NET

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

SuperPanel for ASP.NET Setup Files

The **ComponentOne Studio for ASP.NET** 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 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\<username>\My Documents\ComponentOne Samples

Windows 7/Vista path: C:\Users\<username>\Documents\ComponentOne Samples

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

Common	Contains support and data files that are used by many of the demo programs.
C1WebUi	Contains a Samples folder for the Visual Studio sample project and a readme.txt file.

System Requirements

System requirements for **ComponentOne Studio for ASP.NET** 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
	Visual Studio 2008
	Internet Explorer® 6.0 or later
	Firefox® 2.0 or later
	Safari® 2.0 or later
Disc Drive:	CD or DVD-ROM drive if installing from CD

Note: SuperPanel for ASP.NET requires Microsoft ASP.NET AJAX Extensions installed and a **ScriptManager** on the page before the C1SuperPanel control is placed on the page. You must create an ASP.NET AJAX-Enabled Project so that the **ScriptManager** and Microsoft AJAX Extensions are included on the page. For more information, see Creating an AJAX-Enabled ASP.NET Project (page 14). For more information about Microsoft ASP.NET AJAX Extensions, see http://ajax.asp.net/. For information about the **ScriptManager**, see MSDN.

Uninstalling Super Panel for ASP.NET

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

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

Deploying your Application in a Medium Trust Environment

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

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

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

Modifying or Editing the Config File

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

Edit the Config File

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

1. Locate the medium trust policy file web_mediumtrust.config located by default in the %windir%\Microsoft.NET\Framework\{Version}\CONFIG directory.
2. Open the web_mediumtrust.config file.
3. Add the permissions that you want to grant. For examples, see Adding Permissions (page 5).

Create a Custom Policy Based on Medium Trust

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

1. Locate the medium trust policy file web_mediumtrust.config located by default in the %windir%\Microsoft.NET\Framework\{Version}\CONFIG directory.
2. Copy the web_mediumtrust.config file and create a new policy file in the same directory.

   Give the new a name that indicates that it is your variation of medium trust; for example, AllowReflection_Web_MediumTrust.config.
3. Add the permissions that you want to grant. For examples, see Adding Permissions (page 5).
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 4) 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 4) topic to create or modify a policy file before completing the following.

ReflectionPermission

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

To add reflection permission, complete the following:

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

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

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

```
<NamedPermissionSets>
     <PermissionSet class="NamedPermissionSet" version="1"
Name="ASP.Net">
        <IPermission
            class="ReflectionPermission"
            version="1"
```

```
            Flags="ReflectionEmit,MemberAccess" />
        ...
    </PermissionSet>
</NamedPermissionSets>
```

4. Save and close the web_mediumtrust.config file.

OleDbPermission

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

To add OleDbPermission, complete the following steps:

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

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

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

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

4. Save and close the web_mediumtrust.config file.

FileIOPermission

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

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

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

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

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

```
<NamedPermissionSets>
```

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

4. Save and close the web_mediumtrust.config file.

End-User License Agreement

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

Licensing FAQs

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

What is Licensing?

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

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

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

How does Licensing Work?

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

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

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

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

About This Documentation

Acknowledgements

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

Corporate Headquarters

ComponentOne LLC
201 South Highland Avenue
3rd Floor
Pittsburgh, PA 15206 • USA
412.681.4343
412.681.4384 (Fax)

http://www.componentone.com

ComponentOne Doc-To-Help

This documentation was produced using ComponentOne Doc-To-Help® Enterprise.

Namespaces

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

The general namespace for ComponentOne Web products is **C1.Web.UI.Controls**. The following code fragment shows how to declare a **C1SuperPanel** (which is one of the core **Studio for ASP.NET** classes) using the fully qualified name for this class:

- Visual Basic
```
Dim SuperPanel As C1.Web.UI.Controls.C1SuperPanel
```
- C#
```
C1.Web.UI.Controls.C1SuperPanel SuperPanel;
```

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 C1SuperPanel = C1.Web.UI.Controls.C1SuperPanel
Imports MySuperPanel = MyProject.Objects.C1SuperPanel

Dim sp1 As C1SuperPanel
Dim sp2 As MySuperPanel
```
- C#
```
using C1SuperPanel = C1.Web.UI.Controls.C1SuperPanel;
using MySuperPanel = MyProject.Objects.C1SuperPanel;
```

```
C1SuperPanel sp1;
MySuperPanel sp2;
```

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 SuperPanel for ASP.NET 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 C1SuperPanel control is placed on the page. This allows you to take advantage of ASP.NET AJAX and certain features such as partial-page rendering and client-script functionality of the Microsoft AJAX Library.

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

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

If you are using Visual Studio 2008 and .NET Framework 3.5, you can easily create an AJAX-enabled ASP.NET project without installing separate add-ins because the framework has a built-in AJAX library and controls.

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

The following table summarizes the installations needed:

Visual Studio Version	Additional Installation Requirements
Visual Studio 2008, .NET Framework 3.5	None
Visual Studio 2008 and .NET Framework 2.0 or 3.0 Visual Studio 2005 Service Pack 1	ASP.NET AJAX Extensions 1.0
Visual Studio 2005	ASP.NET AJAX Extensions 1.0 Visual Studio update and add-in (2 installs for Web application project support)

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

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

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

  1. From the File menu, select **New** | Web Site. The New Web Site dialog box opens.
  2. Select .NET Framework 3.5 or the desired framework in the upper right corner. Note that if you choose .NET Framework 2.0 or 3.0, you must install the extensions first.
  3. In the list of templates, select **AJAX 1.0-Enabled ASP.NET 2.0 Web Site**.

4. Click **Browse** to specify a location and then click **OK**.

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

A new AJAX-Enabled Web Site is created at the root of the Web server you specified. In addition, a new Web Forms page called Default.aspx is displayed and a **ScriptManager** control is placed on the form. The **ScriptManger** is needed to enable certain features of ASP.NET AJAX such as partial-page rendering, client-script functionality of the Microsoft AJAX Library, and Web-service calls.

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

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

  1. From the **File** menu, select **New** | Project. The New Project dialog box opens.
  2. Select .NET Framework 3.5 or the desired framework in the upper right corner. Note that if you choose .NET Framework 2.0 or 3.0, you must install the extensions first.
  3. Under **Project Types**, choose either **Visual Basic** or **Visual C#** and then select **Web**. Note that one of these options may be located under **Other Languages**.
  4. Select **AJAX 1.0-Enabled ASP.NET 2.0 Web Application** from the list of **Templates** in the right pane.
  5. Enter a URL for your application in the **Location** field and click **OK**.

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

  A new Web Forms project is created at the root of the Web server you specified. In addition, a new Web Forms page called Default.aspx is displayed and a **ScriptManager** control is placed on the form. The **ScriptManger** is needed to enable certain features of ASP.NET AJAX such as partial-page rendering, client-script functionality of the Microsoft AJAX Library, and Web-service calls.

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

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

  1. From the **File** menu in Microsoft Visual Studio .NET, select **New Web Site**. The **New Web Site** dialog box opens.
  2. Select **ASP.NET AJAX-Enabled Web Site** from the list of Templates.
  3. Enter a URL for your site in the **Location** field and click **OK**.

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

  A new Web Forms project is created at the root of the Web server you specified. In addition, a new Web Forms page called Default.aspx is displayed and a **ScriptManager** control is placed on the form. The **ScriptManger** is needed to enable certain features of ASP.NET AJAX such as partial-page rendering, client-script functionality of the Microsoft AJAX Library, and Web-service calls.

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

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

1. From the **File** menu in Microsoft Visual Studio 2005, select **New Project**. The **New Project** dialog box opens.
2. Under **Project Types**, choose either **Visual Basic Projects** or **Visual C# Projects**. Note that one of these options may be located under **Other Languages**.
3. Select **ASP.NET AJAX-Enabled Web Application** from the list of **Templates** in the right pane.
4. Enter a URL for your application in the **Location** field and click **OK**.

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

A new Web Forms project is created at the root of the Web server you specified. In addition, a new Web Forms page called Default.aspx is displayed and a **ScriptManager** control is placed on the form. The **ScriptManger** is needed to enable certain features of ASP.NET AJAX such as partial-page rendering, client-script functionality of the Microsoft AJAX Library, and Web-service calls.

Adding Super Panel for ASP.NET Component to a Project

When you install **ComponentOne Studio for ASP.NET**, the **Create a ComponentOne Visual Studio 2005\2008 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 2005\2008 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**, the following **Super Panel for ASP.NET** component will appear in the Visual Studio Toolbox customization dialog box:

- C1SuperPanel

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.2.dll. Note that there may be more than one component for each namespace.
5. Click **OK** to close the dialog box.

   The controls are added to the Visual Studio Toolbox.

Adding Studio for ASP.NET Controls to the Form

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

1. Add C1SuperPanel to the Visual Studio toolbox.

2. Double-click the C1SuperPanel control or drag it onto your form.

Note: Expander for ASP.NET requires Microsoft ASP.NET AJAX Extensions installed and a **ScriptManager** on the page before the C1SuperPanel control is placed on the page. You must create an ASP.NET AJAX-Enabled Project so that the **ScriptManager** and Microsoft AJAX Extensions are included on the page. For more information, see Creating an AJAX-Enabled ASP.NET Project (page 14). For more information about Microsoft ASP.NET AJAX Extensions, see http://ajax.asp.net/. For information about the **ScriptManager**, see MSDN.

Adding a Reference to the Assembly

To add a reference to the C1.Web.UI.Conttrols.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 13) for more information.

Key Features

C1SuperPanel includes several unique features, including the following:

- **Rounded Corners**

  Rounded corners appear on C1SuperPanel by default and can be removed by setting the ShowRounder property to **False**.

- **Multiple Scrolling Mode**

  C1SuperPanel provides four scrolling modes: ScrollBar, ScrollButton, ScrollButtonsHover and EdgeHover. Any of the four scrolling modes can be combined to make a mixed mode. See the SuperPanel Scroll Settings (page 33) for more information on the multiple scrolling modes C1SuperPanel supports.

- **Flexible Client Scrolling API Support**

  You can use client side API to scroll to any position of the panel with C1SuperPanel's flexible client scrolling API support.

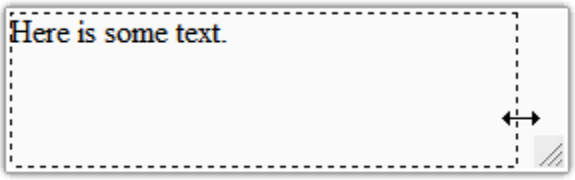
- **Stunning Custom Scroll Bars**

C1SuperPanel provides a custom scrolling bar to take the place of Windows intrinsic scrolling bars. The custom scrolling bars mimic the function of a normal Windows scrolling bar and are synchronized with scrolling position.

- **Resizable**

  C1SuperPanel provides resizing support. The resizing handle will also work as a rounder.

  By default, the AllowResize property is set to True so you are able to resize the SuperPanel at run time. When you resize the C1SuperPanel at run time a dashed rectangular box appears as a visual cue to show you the new size of the SuperPanel before you release the mouse.



- **Scrolling Animation Duration**

  C1SuperPanel includes thirty-one built-in animation transition options that allow you to customize how animation effects are transitioned in the C1SuperPanel control. You can change how the control animates the element scrolling within C1SuperPanel using the AnimationEasing property. By default the AnimationEasing property is set to **EaseLinear** and the control scrolls a smooth linear transition effect.

  For more information see Scrolling Animation (page 35).

- **Customized Content**

  You can add images, text, and controls – standard and third-party controls – to the **C1SuperPanel** control.

- **CSS Styling**

  C1SuperPanel includes CSS supported styling so that you can use cascading style sheets to easily style the **C1SuperPanel** control to match the design of your current Web site.

- **Client-Side Object Model**

  The **C1SuperPanel** control's client-side object model is exposed so that you can easily customize the control with client-side script.

- **Template Support**

  Dynamic Templates can be used in the content area of the panel for achieving rich presentation of the **C1SuperPanel**.

- **Nested SuperPanel**

  You can nest multiple SuperPanels within C1SuperPanel.

- **Browser Support**

  C1SuperPanel includes support for the Internet Explorer (6.0 or later), Firefox (2 or later), and Safari Web browsers.

- **XHTML Compliant**

  C1SuperPanel provides complete XHTML compliance. The output that is generated is fully XHTML 1.1 compliant.

- **Visual Styles**

Use the built-in Visual Styles to quickly change the **C1SuperPanel** control's appearance. Built-in styles include Office 2007 and Vista styles.

SuperPanel for ASP.NET Quick Start

In this quick start you'll explore the functionality of the **C1SuperPanel** control.

Step 1 of 4: Adding C1SuperPanel to the Page

In this step you'll create and set up a Web site and add the **C1SuperPanel** control.

Complete the following steps to add **C1SuperPanel** control to a Web site:

1. Create a new ASP.NET Ajax-Enabled Web site project.

   Note that as you've created an AJAX-Enabled Web site, a **ScriptManager** control initially appears on the page.

2. Click the **Design** tab to enter Design view.

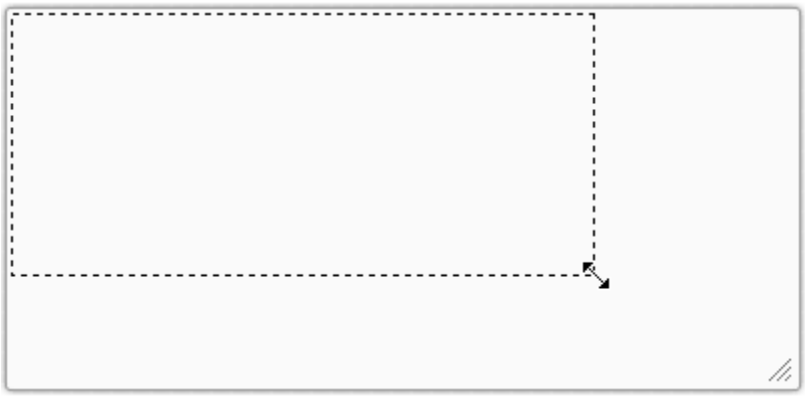
3. In the Visual Studio Toolbox, double-click the **C1SuperPanel** icon to add the control to your page.

   By default, C1SuperPanel's width automatically fills the page. The C1SuperPanel tags are added to your source code.

```
<cc1:C1SuperPanel ID="C1SuperPanel1" runat="server">
</cc1:C1SuperPanel>
```

Run your application and observe:

Click the bottom-right corner of the **C1SuperPanel** control and drag the mouse over the control toward the upper-left corner to reduce the size.



Notice how dashes appear as a visual cue to let you know what the size will be for **C1SuperPanel** once you release the mouse.

In this step you added the C1SuperPanel control to the page and observed its default behaviors at run time. In the next step of the quick start, you'll add content to **C1SuperPanel**.

Step 2 of 4: Adding Content to the C1SuperPanel Control

Adding content to the C1SuperPanel control is as simple as clicking in the body of the control and typing text or adding controls. The following steps assume you've completed the Step 1 of 4: Adding C1SuperPanel to the Page (page 20) topic.

Complete the following steps to add content to **C1SuperPanel**:

1. In Source view add the following markup within the `<div>` tag that appears right after the `</asp:ScriptManager>` tag to create the elements class to get the style for the elements that will appear within the **C1SuperPanel** control.

```
<div class="elements">
<div style="padding: 25px;">
```

2. Add the following tags within the `<cc1:C1Superpanel>` tag to add a button and textboxes to set the **Height** and **Width** properties.

```
Width="300px" Height="300px"
```

3. In Source view add the `<ContentTemplate></ContentTemplate>` tags within the `<cc1:C1SuperPanel></cc1:C1SuperPanel>` tags to create a content template for C1SuperPanel. Within the tags create a class to define the style of the elements within the content template. Your source code should appear like the following:

```
  <ContentTemplate>
     <ul class="elements" style="height: 1011px; width: 1820px;">
<li><p>0</p><a href="#" title="" class="back">go
back</a></li><li><p>1</p><a href="#" title="" class="back">go
back</a></li><li><p>2</p><a href="#" title="" class="back">go
back</a></li><li><p>3</p><a href="#" title="" class="back">go
back</a></li><li><p>4</p><a href="#" title="" class="back">go
back</a></li><li><p>5</p><a href="#" title="" class="back">go
back</a></li><li><p>6</p><a href="#" title="" class="back">go
back</a></li><li><p>7</p><a href="#" title="" class="back">go
back</a></li><li><p>8</p><a href="#" title="" class="back">go
back</a></li><li><p>9</p><a href="#" title="" class="back">go
back</a></li><li><p>10</p><a href="#" title="" class="back">go
back</a></li><li><p>11</p><a href="#" title="" class="back">go
back</a></li><li><p>12</p><a href="#" title="" class="back">go
back</a></li><li><p>13</p><a href="#" title="" class="back">go
back</a></li><li><p>14</p><a href="#" title="" class="back">go
back</a></li><li><p>15</p><a href="#" title="" class="back">go
back</a></li><li><p>16</p><a href="#" title="" class="back">go
back</a></li><li><p>17</p><a href="#" title="" class="back">go
back</a></li><li><p>18</p><a href="#" title="" class="back">go
back</a></li><li><p>19</p><a href="#" title="" class="back">go
back</a></li><li><p>20</p><a href="#" title="" class="back">go
back</a></li><li id="t1"><p>21</p><a href="#" title="" class="back">go
back</a></li><li><p>22</p><a href="#" title="" class="back">go
back</a></li><li><p>23</p><a href="#" title="" class="back">go
back</a></li><li><p>24</p><a href="#" title="" class="back">go
back</a></li><li><p>25</p><a href="#" title="" class="back">go
back</a></li><li><p>26</p><a href="#" title="" class="back">go
back</a></li><li><p>27</p><a href="#" title="" class="back">go
back</a></li><li><p>28</p><a href="#" title="" class="back">go
back</a></li><li><p>29</p><a href="#" title="" class="back">go
back</a></li>
```

```
    </ul>
  </ContentTemplate>
</cc1:C1SuperPanel>
</div>
</div>
```

4. Add the following style markup between the <head> and </head> tags at the top of the document:

```
<style type="text/css">
    .elements ul
    {
        padding: 0px;
        margin: 0px;
    }
    .elements ul li {
    background-color:#DDDDDD;
    border:1px solid black;
    font-weight:bolder;
    height:100px;
    padding:50px;
    position:relative;
    text-align:center;
    width:200px;
    }

    .elements li {
     float:left;
     list-style: none none outside;
    }
    </style>
```

5. In design view notice that the elements within C1SuperPanel's content template are styled.

Run your application and observe:

Notice how horizontal and vertical scrollbars automatically appear since the default value for the Visibility property is "Auto". The style of the scrollbars takes the style of the Arctic visual style since the UseCustomScrollBar property is set to True.

Also notice the horizontal button location appears in the middle and the horizontal scrollbar location appears at the bottom. These are the default settings for the ButtonLocation and ScrollbarLocation properties.

The vertical button location for the vertical scrollbar appears in the center and the vertical scrollbar location appears to the right. These are the default settings for the ButtonLocation and ScrollbarLocation properties.

In this step you added content to C1SuperPanel control. In the next step you'll customize the scroll settings for C1SuperPanel.

Step 3 of 4: Creating a Mixed Scroll Mode in Code

C1SuperPanel provides four scrolling modes: ScrollBar, ScrollButton, ScrollButtonsHover and EdgeHover. Any of the four scrolling modes can be combined to make a mixed mode. In the following steps you will create a mixed scroll mode for C1SuperPanel. The following steps assume you've completed the Step 2 of 4: Adding Content to the C1SuperPanel Control (page 21) topic.

1. Delcare the **C1.Web.UI.Controls.C1SuperPanel** and **C1.Web.UI.Controls.C1SuperPanel.BehaviorSettings** namespace directives in your code file at the top of the page.

   - Visual Basic

```
Imports C1.Web.UI.Controls.C1SuperPanel
Imports C1.Web.UI.Controls.C1SuperPanel.BehaviorSettings
```

- C#

```
using C1.Web.UI.Controls.C1SuperPanel;
using C1.Web.UI.Controls.C1SuperPanel.BehaviorSettings;
```

2. In code file, add the following code to the **Page_Load** event to set the scroll mode of C1SuperPanel. In this case we are using scroll bar mode mixed with scroll button mode.

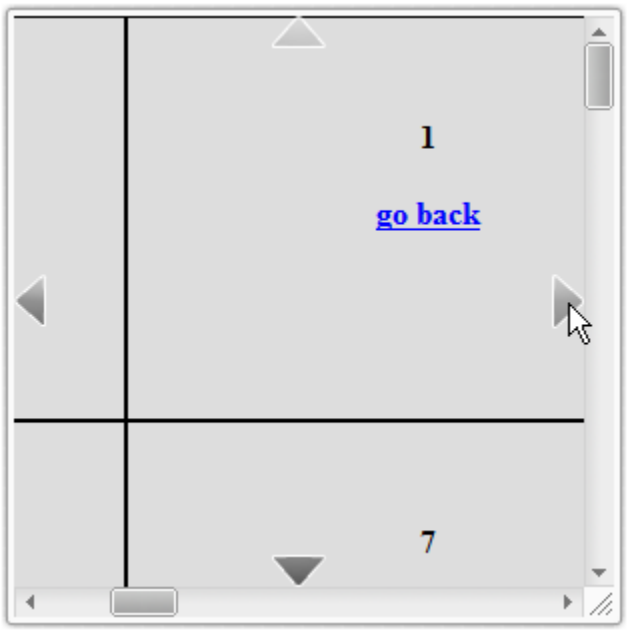
   - Visual Basic

```
Me.C1SuperPanel1.PanelBehaviorSettings.HScroller.Mode =
C1SuperPanelScrollMode.ScrollBar Or C1SuperPanelScrollMode.Buttons
Me.C1SuperPanel1.PanelBehaviorSettings.VScroller.Mode =
C1SuperPanelScrollMode.ScrollBar Or C1SuperPanelScrollMode.Buttons
```

   - C#

```
this.C1SuperPanel1.PanelBehaviorSettings.HScroller.Mode =
C1SuperPanelScrollMode.ScrollBar | C1SuperPanelScrollMode.Buttons;
this.C1SuperPanel1.PanelBehaviorSettings.VScroller.Mode =
C1SuperPanelScrollMode.ScrollBar | C1SuperPanelScrollMode.Buttons;
```

3. Press **F5** to run the application and observe the arrow buttons and scrollbars that appear for the horizontal and vertical scrolling. Also notice when you click or hover over the scroll buttons or use the scrollbars the horizontal or vertical scrolling is performed.



In this step you customized the horizontal and vertical scroll settings of the controls. For more information about the different scroll setting values see SuperPanel Scroll Settings (page 33). In the next step you'll add animation effects to the controls.

Step 4 of 4: Adding Animation Effects to C1SuperPanel

C1SuperPanel includes 31 built-in scrolling animation effects for when you perform horizontal or vertical scrolling on C1SuperPanel. These effects can be set using the AnimationEasing property. In the following steps you will add a bouncing animation effect to the horizontal and vertical scrolling. You will also increase the value of the AnimationDuration property to increase the duration of the scrolling animation. The following steps assume you've completed the Step 3 of 4: Creating a Mixed Scroll Mode in Code (page 22) topic.

Complete the following steps:

1. Select C1SuperPanel on the Web page and then navigate to its properties window.
2. Expand the **PanelBehaviorSettings** node to reveal another list of properties and then complete the following:
   - Set the AnimationEasing property to "EaseinBounce". This property determines the animation transition effect for the scrolling.
   - Set the AnimationDuration property to "1500". This will lengthen the duration of the animation effect, guaranteeing that you will notice the effect when you build the project.
3. Press **F5** to run your application and notice the bouncing effect when you scroll in the **C1SuperPanel** control.

Congratulations, you've completed this quick start guide!

SuperPanel for ASP.NET Top Tips

The following tips will help you when you are using the **C1SuperPanel** control.

Tip1: MaintainScrollPositionOnPostback property

You can set the MaintainScrollPositionOnPostback property of C1SuperPanel to indicate whether to maintain scroll position between post back.

If this property is set to false on each post back the scroll position will reset to 0.

Tip2: Understanding the BubbleScrollingEvent property

The BubbleScrollingEvent property indicates whether to bubble mouse scrolling event to C1SuperPanel parent if the scroller has reached either bound. For example, if we have two nested C1SuperPanels and we are using mouse scrolling to scroll the inner C1SuperPanel, when the inner C1SuperPanel reaches either bound (that is to say the scroller cannot be scrolled anymore) it will test the BubbleScrollingEvent to determine whether it should bubble the mouse scrolling event to its parent which in our case is the outer C1SuperPanel. If the value is true, mouse scrolling will raise the scrolling of the outer C1SuperPanel.

Tip3: Adding keyboard support to C1SuperPanel

C1SuperPanel supports using the keyboard by using specific keys to perform scrolling. To disable key board scrolling, set the KeyboardSupport property to false.

Supported Keys:

The following table describes the functionality of specific keys in the keyboard that can be used for scrolling to areas within C1SuperPanel.

Key	Shif Button Pressed	Effect
Key.Left	True	Scrolls a large change to the left.
Key.Right	True	Scrolls a large change to the right.
Key.Up	True	Scrolls a large change to the top.
Key.Down	True	Scrolls a large change to the bottom.
Key.Left	False	Scrolls a small change to the left.
Key.Right	False	Scrolls a small change to the right.
Key.Up	False	Scrolls a small change to the top.
Key.Down	False	Scrolls to a small change to the bottom.
Key.Home		Scrolls to the minimum value.
Key.End		Scrolls to the maximum value

Tip 4: Understanding the EnableGestureScrolling property

C1SuperPanel can be scrolled by dragging the mouse. The EnableGestureScrolling (default to true) property determines whether this feature is enabled. You can disable this feature by setting this property to false.

Tip 5: Set scroller value at client side

You can change the scroller position by setting the value at client-side using the following script:

```
        function setValue() {
            var sp = $find('C1SuperPanel1');
            // set scroller value
            sp.get_panelBehavior().get_vScroller().set_value(100);
            // repaint scroller
            sp.get_panelBehavior().rePaint();
        }
```

C1SuperPanel Elements

This section provides a visual and descriptive overview of the elements that comprise the C1SuperPanel control.

Content Element

The main part of the C1SuperPanel control is the content area. The content area is a panel.

In the content area you can add rich text through custom HTML content and add arbitrary controls through its content template. Elements in the content area of the control can be added and moved on the control through a simple drag-and-drop operation.

C1SuperPanel includes the following property to make it simple to add and customize any type of item such as text, images and arbitrary controls to the content area:

- ContentTemplate

The following image labels the content area in the C1SuperPanel control:

You can enter text in the content area of the SuperPanel at design time. When you enter text into the content area, C1SuperPanel adds a `<ContentTemplate>` tag inside the `<cc1:C1SuperPanel>` tag like the following:

```
<cc1:C1SuperPanel ID="C1SuperPanel1"
runat="server"VisualStylePath="~/C1WebControls/VisualStyles">
            <ContentTemplate>
                This is where the content information is placed.
            </ContentTemplate>
        </cc1:C1SuperPanel>
```

For more information about creating a content template, see the topic Creating a Content Template (page 48).

Custom ScrollBar Element

C1SuperPanel provides a custom scrolling bar to take the place of Windows intrinsic scrolling bars. The custom scrolling bars mimic the function of a normal Windows scrolling bar and are synchronized with scrolling position.

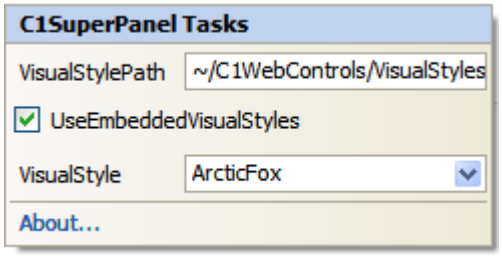
Design-Time Support

The following sections describe how to use **C1SuperPanel** 's design-time environment to configure the C1SuperPanel control.

C1SuperPanel Smart Tag

In Visual Studio, the **C1SuperPanel** control includes a smart tag. A smart tag represents a short-cut tasks menu that provides the most commonly used properties in **C1SuperPanel**.

To access the **C1SuperPanelTasks** menu, click on the smart tag in the upper-right corner of the **C1SuperPanel** control. This will open the **C1SuperPanel Tasks** menu:



The **C1SuperPanel Tasks** menu operates as follows:

- **VisualStylePath**

  The **VisualStylePath** property specifies the location of the visual styles used for the control. By default, embedded visual styles are located in **~/C1WebControls/VisualStyles**. If you create a custom style, add it to this location **~/VisualStyles/StyleName/C1SuperPanel/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 check box is checked by default so that the internal visual styles, such as **ArcticFox** and **Vista** can be used. If you want to use your own custom styles, uncheck this check box and specify the location of your visual styles using the **VisualStylePath** property.

- **Visual Style**

  Clicking the **Visual Style** drop-down box allows you to select from various visual schemes. For more information about available visual styles, see Visual Styles (page 29).

- **About**

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

Super Panel for ASP.NET Appearance

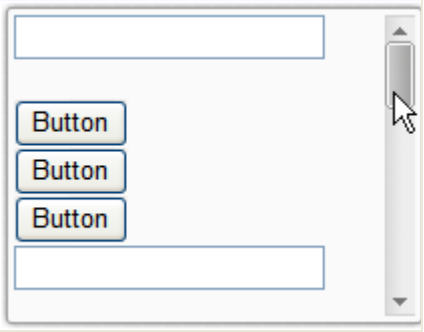
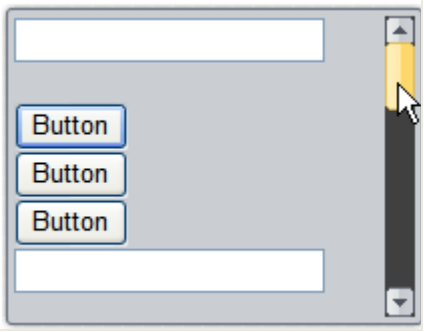
There are several options for customizing the appearance of the C1SuperPanel control. The following sections describe how to change the appearance of the control through built-in Visual Styles as well as how to customize other elements of the C1SuperPanel control.

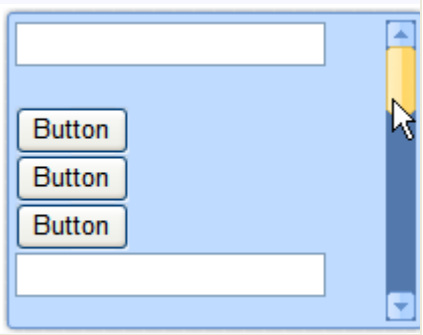
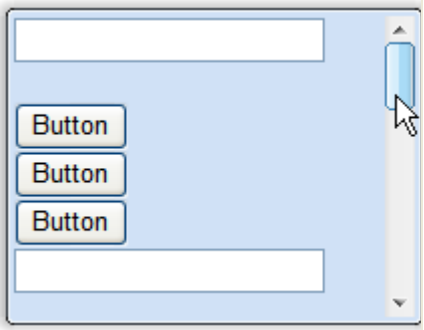
Visual Styles

C1SuperPanel includes visual styles allowing you to easily change the control's appearance. The control includes several built-in visual styles, including Vista and Office 2007 styles, to quickly style your application. You can easily change Visual Styles from the **C1SuperPanel Tasks** menu, from the Properties window, and in code. For more information on changing Visual Styles see the Setting the Visual Style (page 46) topic.

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**, **RanierOrange**, **RanierPurple**, **ShinyBlue**, and **Windows7**. See Custom Visual Styles (page 30) for more information.

The following table illustrates each of the five built-in visual styles included in **Super Panel for ASP.NET**:

Formatting	Appearance
ArcticFox	
Office2007Black	

Office2007Blue	
Office2007Silver	
Vista	

Custom Visual Styles

While **SuperPanel for ASP.NET** comes with five built-in styles, we recognize that there are instances where you will want to customize your C1SuperPanel control. To customize the C1SuperPanel control, 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 style is by modifying one of the control's pre-existing visual styles. You can find the .css sheets and images for **C1SuperPanel**'s visual styles within the installation directory at **C:\Program Files\ComponentOne\Studio for ASP.NET\C1WebUI\VisualStyles**. If you use one the existing style .css sheets, you must change any instances of the default file name, for example **_ArcticFox** to your new style name, for example **_CustomVisualStyle,** in order for your custom style to work.

Before you add your .css file and images, you will have to create a folder to contain them. The folder that will contain them will be buried within a hierarchy of folders. 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 "CustomVisualStyle"), and then, beneath that, create a sub-folder named "C1SuperPanel". The image folder and .css file should be placed underneath the **C1SuperPanel** folder. The folder hierarchy will resemble the following:

This structure of these folders is very important; **C1SuperPanel** will always look for the **~/VisualStyles/StyleName/C1SuperPanel/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 an example on creating a custom visual style, see Adding Custom Visual Styles (page 43).

Content Template

C1SuperPanel provides a content template so you can drag controls like other panel controls to the C1SuperPanel control. You can modify the content template using the ContentTemplate property.

You can enter text in the content area of the SuperPanel at design time. When you enter text into the content area, C1SuperPanel adds a `<ContentTemplate>` tag inside the `<cc1:C1SuperPanel>` tag like the following:

```
<cc1:C1SuperPanel ID="C1SuperPanel1"
runat="server"VisualStylePath="~/C1WebControls/VisualStyles">
            <ContentTemplate>
                This is where the content information is placed.
            </ContentTemplate>
        </cc1:C1SuperPanel>
```

For more information about creating a content template, see the topic Creating a Content Template (page 48).

Rounded Corners

By default, **C1SuperPanel** displays rounded corners as shown below:

The disable rounded corners set the ShowRounder property to **False**.

Here is how the C1SuperPanel control appears without rounded corners:

Drop Shadow

By default, C1SuperPanel displays a drop shadow around the edges of the control as shown below:

To remove the drop shadow set the ShowDropShadow property to **False**.

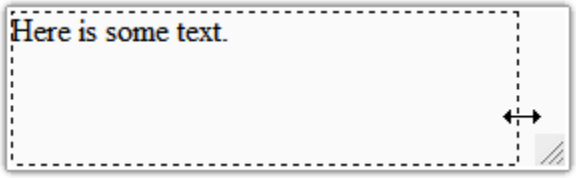
SuperPanel Behavior

The following topics describe the behavior for the C1SuperPanel control.

SuperPanel Sizing

You can specify the initial size of your SuperPanel. To resize the SuperPanel at design time, position your cursor over the bottom-right corner of the SuperPanel, press and hold the left mouse button, drag the SuperPanel edge to resize, and release the mouse when you are done.

By default, the AllowResize property is set to True so you are able to resize the SuperPanel at run time. When you resize the C1SuperPanel at run time a dashed rectangular box appears as a visual cue to show you the new size of the SuperPanel before you release the mouse.



If you want to specify a fixed width and height for SuperPanel you can do so by setting the Height and Width properties and then setting the AllowResize property to **False**.

SuperPanel Custom ScrollBar

By default, C1SuperPanel provides a custom scrolling bar to take the place of Windows intrinsic scrolling bars. The custom scrolling bars mimic the function of a normal Windows scrolling bar and are synchronized with scrolling position.

To disable the custom scrollbar set the UseCustomScrollBar property to False.

SuperPanel Scroll Settings

The C1SuperPanel control contains several scroll settings properties for determining the visibility, button alignment, scroll mode, and scrollbar location for the horizontal and vertical scrollbars on the SuperPanel: **Visibility**, **ScrollButtonAlign**, **ScrollMode**, and **ScrollbarLocation**. These properties are described in the table below.

Property	Enumeration	Description
Visibility	C1ScrollBarVisibility	Gets or sets the visibility option for the superpanel. These options are outlined below: **Hidden**: Scroll bar (button) is always hidden. It's not shown in any case. **Visible**: Scroll bar (button) is always visible and is disabled when not available. **Auto**: Scroll bar (button) is shown when needed
ButtonAlign	C1ScrollButtonAlign	Gets or sets the alignment of the scroll buttons. The buttons can be set to StartSide, **TwoSides**, or **EndSide**. These options are outlined below: **StartSide**: Selecting this option places both buttons on the starting side of the superpanel. **TwoSides**: Selecting this option places buttons on each side of the superpanel. **EndSide**: Selecting this option places both buttons on the ending side of the superpanel.
Mode	C1SuperPanelScrollMode	Gets or sets a value that determines which scroll mode is enabled. You can choose from **Buttons**, **ButtonsHover**, **Edge**, and **ScrollBar**. These options are outlined below: **Buttons**: Selecting this option will place two button images on the control. These buttons must be pressed in order to scroll through the superpanel. **ButtonsHover**: Selecting this option will place two button images on the control. Users can scroll through the SuperPanel by hovering over one of the buttons. **Edge**: Selecting this option enables users to scroll through the SuperPanel by hovering over SuperPanel's inner edge to do the scrolling. **ScrollBar**: Selecting this option enables users to scroll through the SuperPanel by using the scrollbars. Any of these options may be combined to create a mixed scrolling mode.
(Horizontal)ScrollbarLocation	C1HScrollBarPosition	Gets or sets the scroll position option for the horizontal scrollbar. This property will determine the position of the scrollable superpanel at run-time.

(Vertical)ScrollbarLocation	C1VScrollBarPosition	Gets or sets the scroll position option for the vertical scrollbar. This property will determine the position of the scrollable superpanel at run-time.

Sample Project Available

For a demonstration of client-side functions in the C1SuperPanel control, see the **ScrollMode** page of the **ControlExplorer** sample.

Scrolling Elements

You can specify what and where to scroll elements within **SuperPanel** using the **ScrollTo**, **ScrollToChild**, and **ScrollOffset** methods.

You can use any of the following ways to specify the target position:

- A hash {top:x, left:y}, x and y can be any kind of number string
- The string "max" to scroll to the end
- A string such as "100", "100px", "+=15px", etc
- A child element of the scrollable element

Scrolling to a Location within SuperPanel

You can specify what to scroll by simply calling the name of the object or element you want to be scrolled. For example you can use the **ScrollTo** method for scrolling to a specific location in the SuperPanel.

```
<script language="javascript" type="text/javascript">
        function scrollTo() {
            var scroll = $find('<%=C1SuperPanel1.ClientID %>');
            scroll.get_panelBehavior().scrollTo(300, 200);
        }

    </script>
```

Scrolling to a child element within SuperPanel

You can use the **scrollToChild** method to scroll to a child element in the SuperPanel.

```
<script language="javascript" type="text/javascript">
        function scrollToChild() {
            var scroll = $find('<%=C1SuperPanel1.ClientID %>');
            var d = $get('t1');
            d.style.border = "1px solid red";
            scroll.get_panelBehavior().scrollToChild(d);
        }
    </script>
```

Scrolling to offset parameters

You can use the **scrollOffset** method to scroll the SuperPanel to a specified position along the XAxis, YAxis or both based on the specified offset parameters.

```
<script language="javascript" type="text/javascript">
```

```
        function scrollOffSet() {
            var scroll = $find('<%=C1SuperPanel1.ClientID %>');
            scroll.get_panelBehavior().scrollOffset(100, 100);
        }
    </script>
```

Scrolling Animation

The following topics detail C1SuperPanel's scrolling animation effects and its animation duration.

Scrolling Animation Transitions

C1SuperPanel includes thirty-one built-in animation transition options that allow you to customize how animation effects are transitioned in the C1SuperPanel control. You can change how the control animates the element scrolling within C1SuperPanel using the AnimationEasing property. By default the AnimationEasing property is set to **EaseLinear** and the control scrolls a smooth linear transition effect.

Sample Project Available

For a demonstration of each transition effect possible in the C1SuperPanel control, see the **Animation** page of the **ControlExplorer** sample.

The following table describes each transition effect choice:

Transition Name	Transition Description
EaseLinear(default)	Linear easing. Moves smoothly from start to end without acceleration or deceleration
EaseOutElastic	Elastic easing out. Begins quickly and then decelerates.
EaseInElastic	Elastic easing in. Begins slowly and then accelerates.
EaseInOutElastic	Elastic easing in and out. Begins slowly, accelerates halfway, and then decelerates.
EastOutBounce	Bouncing easing out. Begins quickly and then decelerates. The number of bounces is related to the duration: longer durations produce more bounces.
EaseInBounce	Bouncing easing in. Begins slowly and then accelerates. The number of bounces is related to the duration: longer durations produce more bounces.
EaseInOutBounce	Bouncing easing in and out. Begins slowly, accelerates until halfway, and then decelerates.
EaseOutExpo	Exponential easing out. Begins quickly and then decelerates.
EaseInExpo	Exponential easing in. Begins slowly and then accelerates.
EaseInOutExpo	Exponential easing in and out. Begins slowly, accelerates until halfway, and then decelerates.
EaseOutQuad	Quadratic easing out. Begins at full velocity then decelerates to zero.

EaseInQuad	Quadratic easing in. Begins at zero velocity then slowly accelerates.
EaseInOutQuad	Quadratic easing in and out. Begins slowly, accelerates until halfway, and then decelerates to zero velocity again.
EaseOutSine	Sinusoidal easing out. Begins quickly and then decelerates.
EaseInSine	Sinusoidal easing in. Begins slowly and then accelerates.
EaseInOutSine	Sinusoidal easing in and out. Begins slowly, accelerates until halfway, and then decelerates.
EaseInOutCirc	Circular easing in and out. Begins slowly, accelerates until halfway, and then decelerates.
EaseOutCirc	Circular easing out. Begins quickly and then decelerates.
EaseInCirc	Circular easing in. Begins slowly and then accelerates.
EaseInOutCubic	Cubic easing in and out. Begins at zero velocity, accelerates until halfway, and then decelerates to zero velocity again.
EaseInCubic	Cubic easing in. Begins at zero velocity and then accelerates.
EaseOutCubic	Cubic easing in and out. Begins at full velocity and then decelerates to zero.
EaseInQuint	Quintic easing in. Begins at zero speed then accelerates.
EaseInOutQuint	Quintic easing in and out. Begins at zero speed, accelerates until halfway, and then decelerates to zero
EaseOutQuint	Quintic easing out. Begins at full velocity and then decelerates to zero.
EaseOutBack	Back easing out. Begins quickly and then decelerates.
EaseInBack	Back easing in. Begins slowly and then accelerates.
EaseInOutBack	Back easing in and out. Begins slowly, accelerates until halfway, and then decelerates.
EaseOutQuart	Quartic easing out. Begins quickly and then decelerates.
EaseInQuart	Quartic easing in. Begins slowly and then accelerates.
EaseInOutQuart	Quartic easing in and out. Begins at zero velocity, accelerates until halfway, and then decelerates to zero velocity again.

Scrolling Animation Duration

You can set how long each C1SuperPanel scrolling animation effect takes by using the AnimationDuration property. The unit of time used for specifying animation effect duration is in milliseconds, and the default setting for the AnimationDuration property is **500** milliseconds (so half a second). Increase this value for a longer animation effect, and decrease this number for a shorter animation effect.

Client-Side Functionality

C1SuperPanel includes a robust client-side object model, where a majority of server-side properties can be set on the client-side and client-side events are described in the properties window.

When a C1SuperPanel control is rendered, an instance of the client side control will be created automatically. This means that you can enjoy the convenience of accessing properties and methods of the C1SuperPanel control without having to postback to the server.

For example, suppose a C1SuperPanel control with name **C1SuperPanel1** is hosted on Web page; when the page is rendered, a corresponding client side object will be created. Use the following syntax to get the client object:

```
$get("C1SuperPanel1").control
```

OR

```
$find("C1SuperPanel1")
```

By using **C1SuperPanel**'s client-side functionality, you can implement many features in your Web page without the need to take up time by sending information to the Web server. Thus, using client-side methods and events can increase the efficiency of your Web site.

Sample Project Available

For a demonstration of client-side functions in the C1SuperPanel control, see the **ClientSide** page of the **ControlExplorer** sample.

Client-Side Properties

The following conventions are used when accessing 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 example in the code below the C#, Visual Basic, and JavaScript examples are equivalent:

- Visual Basic

```
Dim s As String = C1SuperPanel1.AccessKey
C1SuperPanel1.AccessKey = s
```

- C#

```
string s = C1SuperPanel1.AccessKey;
C1SuperPanel1.AccessKey = s;
```

- JavaScript

```
var SuperPanel = $get("C1SuperPanel1").control;
var s = SuperPanel.get_accessKey();
SuperPanel.set_accessKey(s);
```

For an example of setting C1SuperPanel's **Height** and **Width** properties at run time, see Set the Height and Width at Run Time (page 52).

C1SuperPanel includes a rich client-side object model in which several properties can be set on the client side. For information about these client-side methods and what properties can be set on the client side, see the C1SuperPanel Reference.

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

Client Side Event table

Event Server-Side Property Name	Event Name	Description
OnClientInitialized	ClientInitialized	Gets or sets the name of the client-side function to all when initialized
OnClientResized	ClientResized	Gets or sets the on client-side function to all when C1SuperPanel is resized.
OnClientScrolled	ClientScrolled	Gets or sets the name of the client-side function to all when scrolling ends.
OnClientScrolling	ClientScrolling	Gets or sets the name of the client-side function to all when scrolling ends.

Sample Project Available

For a demonstration of client-side functions in the C1SuperPanel control, see the **ClientSide** page of the **ControlExplorer** sample.

Using the C1SuperPanelExtender Control

The **C1SuperPanelExtender** is an extender control that uses the C1SuperPanel control to wrap any ASP.NET control.

Any ASP.NET control can be wrapped using C1SuperPanelExtender's **TargetControlID** property. The **TargetControlID** property gets or sets the ID of the control that the extender is associated with.

The **C1SuperPanelExtender** control includes the following properties:

- EmbeddedVisualStyles property – Returns a string array with embedded visual style names.
- Height property – Gets or sets the height.
- PanelBehaviorSettings property – Gets the panel behavior settings for the C1SuperPanelExtender control.
- VisualStyle property – Gets or sets the visual style name used by the control.
- Width property – Gets or sets the width.

To see how to use the **C1SuperPanelExtender** control to wrap an ASP.NET control, see Wrapping an ASP.NET Control Using C1SuperPanelExtender (page 42).

SuperPanel for ASP.NET 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.

Note: The ComponentOne Samples are also available at http://helpcentral.componentone.com/Samples.aspx.

C# Sample

The following pages within the **ControlExplorer** sample installed with **ComponentOne Studio for ASP.NET** detail the C1SuperPanel control's functionality:

Sample	Description
VisualStyles	Displays the built-in C1SuperPanel control Visual Styles including ArcticFox, Office2007Black, Office2007Blue, Office2007Silver, and Vista.
Animation	Displays various animation effects as well as expand and collapse transitions, durations, and delays.
Orientation	Shows how to use C1SuperPanel's ContentTemplate property to create images and it also shows how to use C1SuperPanel's Visibility property to determine when to display horizontal and vertical scrollbars on C1SuperPanel.
ClientSide	Demonstrates how to use the different **ScrollTo** methods for specifying what to scroll to such as scrolling to a location, a child, or an offset position within C1SuperPanel.
EventHandler	Demonstrates how to use the following client functions for C1SuperPanel: **OnClientInitialized**, **OnClientScrolled**, and **OnClientScrolling**.
ContentNavigation	Shows how to use C1SuperPanel's ContentTemplate property to style create elements such as links, paragraphs, and headings.
ScrollMode	Shows how the various types of SuperPanel scroll modes behave and appear. The scroll modes include the following: scrollbar, scroll buttons, system scrollbar, hover edge scrolling, and mixed scrolling.

SuperPanel for ASP.NET Task-Based Help

The task-based help assumes that you are familiar with programming in ASP.NET and know how to use controls in general. By following the steps outlined in the help, you will be able to create projects demonstrating a variety of C1SuperPanel's features, and get a good sense of what the C1SuperPanel control can do.

Each topic provides a solution for specific tasks using the C1SuperPanel control. Each task-based help topic also assumes that you have created a new ASP.NET AJAX-Enabled project and added references to the

appropriate assemblies. For additional information on this topic, see Creating an AJAX-Enabled ASP.NET Project.

Note: SuperPanel for ASP.NET requires Microsoft ASP.NET AJAX Extensions installed and a **ScriptManager** on the page before the C1SuperPanel control is placed on the page. You must create an ASP.NET AJAX-Enabled Project so that the **ScriptManager** and Microsoft AJAX Extensions are included on the page. For more information about Microsoft ASP.NET AJAX Extensions, see http://ajax.asp.net/. For information about the **ScriptManager**, see MSDN.

Creating a C1SuperPanel in Code

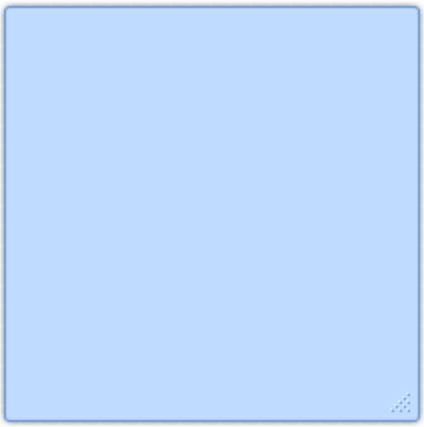
Creating a C1SuperPanel control in code is fairly simple. In the following steps you'll add a **PlaceHolder** control to the page, declare the namespace, customize the C1SuperPanel, and add the control to the **PlaceHolder**.

Complete the following steps:

1. In the Solution Explorer, right-click on your project name and select **Add Reference**. The **Add Reference** dialog box appears.
2. In the **Add Reference** dialog box locate the C1.Web.UI.Controls.2.dll and click **OK** to add the C1.Web.UI.Controls.2.dll reference to your project.
3. In Design view, navigate to the Visual Studio Toolbox and add a **PlaceHolder** control to your page.
4. Delcare the **C1.Web.UI.Controls.C1SuperPanel** namespace directive in your code file at the top of the page.
   - Visual Basic
     ```
     Imports C1.Web.UI.Controls.C1SuperPanel
     ```
   - C#
     ```
     using C1.Web.UI.Controls.C1SuperPanel;
     ```
5. Add the following code to the **Page_Load** event to create and customize the **C1SuperPanel** control.
   - Visual Basic
     ```
     ' Create a new C1SuperPanel.
     Dim C1SP As New C1SuperPanel
     ' Set the control's size, appearance, and content.
     C1SP.VisualStyle = "Office2007Blue"
     C1SP.Height = 200
     C1SP.Width = 200
     ' Add the C1SuperPanel to the PlaceHolder control.
     PlaceHolder1.Controls.Add(C1SP)
     ```
   - C#
     ```
     // Create a new C1SuperPanel.
     C1SuperPanel C1SP = new C1SuperPanel();
     // Set the control's size, appearance, and content.
     C1SP.VisualStyle = "Office2007Blue";
     C1SP.Height = 200;
     C1SP.Width = 200;
     // Add the C1SuperPanel to the PlaceHolder control.
     PlaceHolder1.Controls.Add(C1SP);
     ```

Run your application and observe:

The C1SuperPanel control appears on your page:



Disabling Vertical Scrolling

You can disable vertical scrolling by setting the **VScroller.Visibility** property to None.

This topic shows how to modify the **VScroller.Visibility** property in Design view and in Source view.

In Source View

In Source view add `VScroller Visibility="None"` to the `<PanelBehaviorSettings>` tag so it appears similar to the following:

```
<cc1:C1SuperPanel ID="C1SuperPanel1" runat="server"
UseEmbeddedVisualStyles="True"
            VisualStyle="ArcticFox"
VisualStylePath="~/C1WebControls/VisualStyles"
            Width="701px">
            <PanelBehaviorSettings>
                <VScroller Visibility="None">
                </VScroller>
            </PanelBehaviorSettings>
        </cc1:C1SuperPanel>
```

From the Properties Window

You can set the **Visibility** property to None from the Properties window:

1. Click on the **C1SuperPanel** to select it.
2. Navigate to the Properties window and expand the **PanelBehaviorSettings** node.
3. Locate the **VScroller** node and expand it.
4. Select the dropdown arrow next to the **Visibility** property and select None.

Wrapping an ASP.NET Control Using C1SuperPanelExtender

You can wrap any ASP.NET control such as a **Label** control in the **C1SuperPanel** control using the **TargetControlID** property.

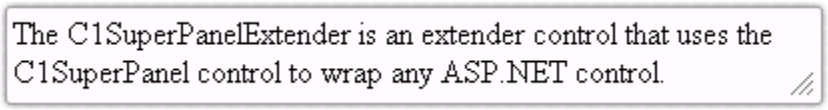
This topic shows how to wrap any ASP.NET control such as a **Label** control at design time.

1. Add an ASP.NET **Label** control and a **C1SuperPanelExtender** control to your page.
2. Set the Label's **Text** property to the following: "The C1SuperPanelExtender is an extender control that uses the **C1SuperPanel** control to wrap any ASP.NET control."
3. Click on the **C1SuperPanelExtender** control to select it.

4. Navigate to the Properties window and set the **TargetControlID** property to “Label1”.
5. Press F5 to run your project.

This task illustrates the following:

The ASP.NET **Label** control is wrapped in the **C1SuperPanel** control.



Customizing C1SuperPanel's Appearance

The following topics show how to change C1SuperPanel’s appearance.

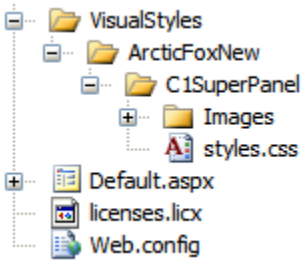
Adding Custom Visual Styles

You can use the **VisualStyle**, **VisualStylePath**, and **UseEmbeddedVisualStyles** properties to create a custom visual style for your C1SuperPanel.

For more information on custom visual styles, see Custom Visual Styles (page 30).

To create a custom visual style for the C1SuperPanel control based on the **Arctic** visual style, complete the following:

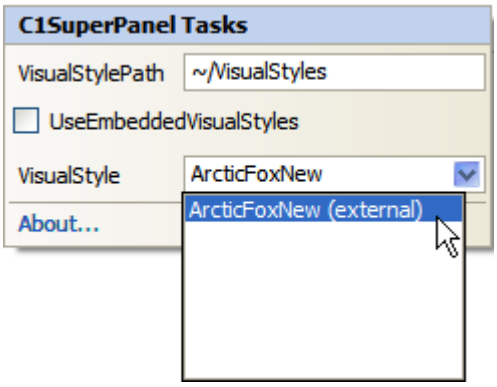
1. Copy the theme folder **C:\Program Files\ComponentOne\Studio for ASP.NET\C1WebUI\VisualStyles\ArcticFox\C1SuperPanel** to a new folder in your Visual Studio project. ~/VisualStyles/ArcticFoxNew/C1SuperPanel.



2. Open the styles.css file inside the new folder, rename all the class names to make sure they end with the new visual style name "ArcticFoxNew". Replace all instances of **_ArcticFox** with **_ArcticFoxNew**.

   Suppose the original class name is **.C1SuperPanel_ArcticFox**, then rename it to **.C1SuperPanel_ArcticFoxNew**.
3. In the styles.css file find the ".C1spnl-ArcticFoxNew" and replace it with the lower-case embedded visual style name, ".C1spnl-arcticfoxnew".
4. In the styles.css file find the ".spnl-ArcticFoxNew" and replace it with the lower-case embedded visual style name, ".spnl-arcticfoxnew".
5. Save and close the modified .css file.
6. Select the **Refresh** button in your project’s solution explorer to add the new styles.css file to your project.
7. Add the **C1SuperPanel** control to your page.
8. Select the C1SuperPanel control on your page and navigate to its properties window and set the **VisualStylePath** property to ~/VisualStyles.

9. Set the **UseEmbeddedVisualStyles** property to False.
10. Set the **VisualStyle** property to the new external custom theme, ArcticFoxNew (external).



11. Now you can open the new styles.css file and edit the corresponding CSS classes. In this example, we modify the border color and border width for the SuperPanel control. Locate the .C1SuperPanel_ArcticFoxNew selector and modify the border color to "#666699" and border width to "10px".

```
.C1SuperPanel_ArcticFoxNew
{
    background: #fafafa url('Images/resizer.png') no-repeat right bottom;
    display:inline-block;
    -moz-outline:0 none;
    outline:0;
    border: solid 10px #666699;
}
```

12. Rebuild your project and observe the new visual style changes to C1SuperPanel.

This task illustrates the following:

The following image shows how the C1SuperPanel control appears after you run the program:

Applying the Visual Style to Children Elements in the SuperPanel

You can apply the same visual style you used for the **C1SuperPanel** control to all of your child elements in the **C1SuperPanel** control. This will make your UI have the same look and feel.

You can easily apply the visual style to children elements in Source view, from the Properties window, and in code.

In Source View

In Source view add `ApplyVisualStyleToChildren="True"` and `VisualStyle="Office2007Black"` to the `<cc1:C1SuperPanel>` tag. Also add `<ContentTemplate></ContentTemplate>` tags to include the **C1Menu** control in the **C1SuperPanel** so it appears similar to the following:

```
<cc1:C1SuperPanel ID="C1SuperPanel1" runat="server"
            ApplyVisualStyleToChildren="True" Height="185px"
UseEmbeddedVisualStyles="True"
            VisualStyle="Office2007Black"
VisualStylePath="~/C1WebControls/VisualStyles"
            Width="701px">
            <ContentTemplate>
                <cc2:C1Menu ID="C1Menu1" runat="server"
UseEmbeddedVisualStyles="True"
                    VisualStyle="ArcticFox"
VisualStylePath="~/C1WebControls/VisualStyles">
                    <Items>
                        <cc2:C1MenuItem runat="server"
Text="LinkItem1">
                        </cc2:C1MenuItem>
                        <cc2:C1MenuItem runat="server"
Text="LinkItem2">
                            <Items>
                                <cc2:C1MenuItem runat="server"
Text="LinkItem1">
                                </cc2:C1MenuItem>
                                <cc2:C1MenuItem runat="server"
Text="LinkItem2">
                                </cc2:C1MenuItem>
                                <cc2:C1MenuItem runat="server"
Text="LinkItem3">
                                </cc2:C1MenuItem>
                                <cc2:C1MenuItem runat="server"
Text="LinkItem4">
                                </cc2:C1MenuItem>
                            </Items>
                        </cc2:C1MenuItem>
                        <cc2:C1MenuItem runat="server"
Text="LinkItem3">
                            <Items>
                                <cc2:C1MenuItem runat="server"
Text="LinkItem1">
                                </cc2:C1MenuItem>
                                <cc2:C1MenuItem runat="server"
Text="LinkItem2">
                                </cc2:C1MenuItem>
                            </Items>
                        </cc2:C1MenuItem>
                        <cc2:C1MenuItem runat="server"
Text="LinkItem4">
                        </cc2:C1MenuItem>
                    </Items>
                </cc2:C1Menu>
            </ContentTemplate>
            <PanelBehaviorSettings>
                <VScroller Visibility="None">
                </VScroller>
            </PanelBehaviorSettings>
        </cc1:C1SuperPanel>
```

In Design View

You can modify the ApplyVisualStyleChildren property from the Properties window:

1. Add the **C1SuperPanel** control to your page.
2. Drag the **C1Menu** icon to the **C1SuperPanel** control so it appears within the **C1SuperPanel** control.
3. Added some items to the **C1Menu** control.
4. Click on the **C1SuperPanel** to select it.
5. Navigate to the Properties window and select the drop-down arrow next to the **VisualStyle** property.
6. Select a style to apply, for example **Vista**.
7. Locate the ApplyVisualStyletoChildren property and set it to True.

This task illustrates the following:

The child control, **C1Menu**, has the same visual style as the **C1SuperPanel** control to make it have the same look and feel.

Setting the Visual Style

The control includes several built-in visual styles, including Vista and Office 2007 styles, to style your application. For more information about available styles, see Visual Styles (page 29). You can easily change Visual Styles in Source view, from the **C1SuperPanel Tasks** menu, from the Properties window, and in code.

In Source View

In Source view add `VisualStyle="Vista"` to the `<cc1:C1SuperPanel>` tag so it appears similar to the following:

```
<cc1:C1SuperPanel ID="C1SuperPanel1" runat="server"
UseEmbeddedVisualStyles="True" VisualStyle="Vista">
<PanelBehaviorSettings Enabled="True" showrounder="True">
<HScroller SmallChange="1" LargeChange="0" Value="0" Maximum="100"
Minimum="0" IncreaseButtonOffsetX="0" DecreaseButtonOffsetX="0"
IncreaseButtonOffsetY="0" DecreaseButtonOffsetY="0"></HScroller>

<VScroller SmallChange="1" LargeChange="0" Value="0" Maximum="100"
Minimum="0" IncreaseButtonOffsetX="0" DecreaseButtonOffsetX="0"
IncreaseButtonOffsetY="0" DecreaseButtonOffsetY="0"></VScroller>
</PanelBehaviorSettings>
</cc1:C1SuperPanel>
```

From the Tasks Menu

You can access the **VisualStyle** property from the **C1SuperPanel Tasks** menu:

1. Click on the **C1SuperPanel**'s smart tag to open the **C1SuperPanel Tasks** menu.
2. Click the **Visual Style** drop-down arrow and select a Visual Style, for example **Vista**.

   The Visual Style you choose will be applied to the **C1SuperPanel**.

From the Properties Window

You can select a Visual Style to apply from the Properties window:

1. Click on the **C1SuperPanel** to select it.
2. Navigate to the Properties window and select the drop-down arrow next to the **VisualStyle** property.
3. Select a style to apply, for example **Vista**.

   The Visual Style you chose will be applied to the **C1SuperPanel**.

In Code

Add the following code to the **Page_Load** event to set the **VisualStyle** property to **Vista**:

- Visual Basic

```
Me.C1SuperPanel1.VisualStyle = "Vista"
```

- C#

```
this.C1SuperPanel1.VisualStyle = "Vista";
```

This task illustrates the following:

The following image shows a C1SuperPanel with the **Vista** visual style:

Changing the Font

To further customize your application, you can easily set the font of the **C1SuperPanel** to be similar to the font selected throughout your application. You can easily change the font in Source view, from the Properties window, and in code.

In Source View

In Source view add text to the `<cc1:C1SuperPanel>` tag so it appears similar to the following:

```
<cc1:C1SuperPanel ID="C1SuperPanel1" runat="server"
            style="top: 0px; left: 0px; height: 85px" Font-Bold="True"
            Font-Names="Arial" Font-Size="Small" ForeColor="Red"
            VisualStylePath="~/C1WebControls/VisualStyles">
            <ContentTemplate>
                Here is red text.
            </ContentTemplate>
```

When you run your application, the text in the body of the **C1SuperPanel** control will now appear small, bold, red, and in the Arial font.

In Code

Add the following code to the **Page_Load** event to customize the font's appearance:

- Visual Basic

```
Me.C1SuperPanel1.Font.Bold = True
Me.C1SuperPanel1.Font.Name = "Arial"
Me.C1SuperPanel1.Font.Size = FontUnit.Small
Me.C1SuperPanel1.ForeColor = Color.Red
```

- C#

```
this.C1SuperPanel1.Font.Bold = true;
```

```
this.C1SuperPanel1.Font.Name = "Arial";
this.C1SuperPanel1.Font.Size = FontUnit.Small;
this.C1SuperPanel1.ForeColor = Color.Red;
```

Note: The **System.Drawing** namespace must be imported for the above code to work correctly.

When you run your application, the text in the body of the **C1SuperPanel** control will now appear small, bold, red, and in the Arial font.

Creating a Content Template

Content templates are useful for customizing your SuperPanel for your application and for displaying additional information in the content area in the panel.

In the following example, you will add a control to SuperPanel's content template in design view and in source view.

In Design View

In Design view drag and drop a few controls onto the C1SuperPanel control and then go to Source view to see the controls added within the `<ContentTemplate> </ContentTemplate>` tags.

It will appear similar to the following:

```
<cc1:C1SuperPanel ID="C1SuperPanel1" runat="server"
            VisualStylePath="~/C1WebControls/VisualStyles">
            <ContentTemplate>
                <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
                <cc3:C1Calendar ID="C1Calendar1" runat="server"
height="180px" width="230px">
                </cc3:C1Calendar>
            </ContentTemplate>
<PanelBehaviorSettings Enabled="True">
<HScroller SmallChange="1" LargeChange="0" Value="0" Maximum="100"
Minimum="0" IncreaseButtonOffsetX="0" DecreaseButtonOffsetX="0"
IncreaseButtonOffsetY="0" DecreaseButtonOffsetY="0"></HScroller>

<VScroller SmallChange="1" LargeChange="0" Value="0" Maximum="100"
Minimum="0" IncreaseButtonOffsetX="0" DecreaseButtonOffsetX="0"
IncreaseButtonOffsetY="0" DecreaseButtonOffsetY="0"></VScroller>
</PanelBehaviorSettings>
        </cc1:C1SuperPanel>
```

In Source View

In Source view add text to the `<cc1:C1SuperPanel>` tag so it appears similar to the following:

```
<cc1:C1SuperPanel ID="C1SuperPanel1" runat="server"
            VisualStylePath="~/C1WebControls/VisualStyles">
```

```
            <ContentTemplate>
                <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
                <cc3:C1Calendar ID="C1Calendar1" runat="server"
height="180px" width="230px">
                </cc3:C1Calendar>
            </ContentTemplate>
<PanelBehaviorSettings Enabled="True">
<HScroller SmallChange="1" LargeChange="0" Value="0" Maximum="100"
Minimum="0" IncreaseButtonOffsetX="0" DecreaseButtonOffsetX="0"
IncreaseButtonOffsetY="0" DecreaseButtonOffsetY="0"></HScroller>

<VScroller SmallChange="1" LargeChange="0" Value="0" Maximum="100"
Minimum="0" IncreaseButtonOffsetX="0" DecreaseButtonOffsetX="0"
IncreaseButtonOffsetY="0" DecreaseButtonOffsetY="0"></VScroller>
</PanelBehaviorSettings>
        </cc1:C1SuperPanel>
```

Setting the Content Background Color

You can customize the appearance of the C1SuperPanel by using styles. You can create a new style front style from scratch or you can modify its existing visual styles. In this topic you'll customize the **ArcticFox** visual style by adding a style to change the background color of the control. The following topic assumes you've added a C1SuperPanel to the page.

Complete the following steps:

1. Click on the **C1SuperPanel**'s smart tag to open the **C1SuperPanel Tasks** menu.
2. From the Tasks menu, click the **VisualStyle** drop-down arrow and select the **ArcticFox** visual style. The style you'll add in the next steps will be added on top of this theme.
3. Switch to Source view.
4. Add the following style markup between the `<head>` and `</head>` tags at the top of the document:

```
<style>
.C1SuperPanel_ArcticFox {
   background: #99CCCC;;
   display:block;
   margin:0;
   padding:0;
}
</style>
```

   This will set the background color of the control's content area.

5. Press **F5** to run your project.

   The **C1SuperPanel** control will appear in the **ArcticFox** theme and the control's content area will appear with a green background:

Setting C1SuperPanel Behaviors

The C1SuperPanel control has a list of properties that affect how the control behaves at run time. Some of the properties affect how the control acts when loaded, whereas others affect the users' interactions with the control. The following topics will instruct you on how to modify the run-time actions of the control.

Using the System ScrollBars on SuperPanel

To use the Windows intrinsic scrolling bars on C1SuperPanel rather than its custom scrolling bars, set UseCustomScrollBar property to False.

Adding Keyboard Support

The **C1SuperPanel** control lets you easily add keyboard accessibility to the control. You can use **AccessKey** property to set how the user navigates to the control and through your user interface.

In the following examples you'll set the **AccessKey** property to **s** so that pressing the ALT+S key combination at run time brings the **C1SuperPanel** control into focus.

In Source View

In Source view add `AccessKey="s"` to the `<cc1:C1SuperPanel>` tag so it appears similar to the following:

```
<cc1:C1SuperPanel ID="C1SuperPanel1" runat="server" style="top: 0px; left: 0px; height: 85px" AccessKey="s">
```

In Design View

In Design view, select the **C1SuperPanel** control and in the Properties window set the **AccessKey** property to "**s**".

In Code

Add the following code to the **Page_Load** event to set the **AccessKey** property to "**s**":

- Visual Basic

```
Me.C1SuperPanel1.AccessKey = "s"
```

- C#

```
this.C1SuperPanel1.AccessKey = "s";
```

Animate Scrolling Within an Element

C1SuperPanel contains thirty-one transition effects that allow you to customize interaction with the control. In this topic, you will set the AnimationEasing and AnimationDuration properties to create an animation effect that occurs when you scroll elements within the C1SuperPanel. This topic illustrates how to set each of these properties in Design view, in Source view, and in code.

Setting Animation Effects in Design View

Complete the following steps to set animation effects:

1. Drag and drop a **C1Calendar** control inside the **C1SuperPanel** control.
2. Select C1SuperPanel on the Web page and then navigate to the Properties window.
3. Expand the **PanelBehaviorSettings** node to reveal another list of properties and then complete the following:
    - Set the AnimationEasing property to **EaseOutCirc**. This property determines the animation transition effect.
    - Set the AnimationDuration property to **1500**. This will lengthen the duration of the animation effect, guaranteeing that you will notice the effect when you build the project.

4. Press **F5** to run your project and then click on any date in the month view area of the C1Calendar control. Observe that it moves diagonally to the lower right corner of the page before settling into a resting state.

Setting Animation Effects in Source View

In Source view place `<PanelBehaviorSettings Enabled="True" animationduration="5000" animationeasing="EaseOutCirc"/>` between the `<cc1:C1SuperPanel>` and `</cc1:C1SuperPanel>` tags so that the markup appears similar to the following:

```
<cc1:C1SuperPanel ID="C1SuperPanel1" runat="server" Height="1129px"
            VisualStylePath="~/C1WebControls/VisualStyles">
            <ContentTemplate>
                <cc3:C1Calendar ID="C1Calendar1" runat="server"
height="180px" width="230px">
                </cc3:C1Calendar>
            </ContentTemplate>
<PanelBehaviorSettings Enabled="True" animationduration="5000"
                animationeasing="EaseOutCirc">
<HScroller SmallChange="1" LargeChange="0" Value="0" Maximum="100"
Minimum="0" IncreaseButtonOffsetX="0" DecreaseButtonOffsetX="0"
IncreaseButtonOffsetY="0" DecreaseButtonOffsetY="0"></HScroller>

<VScroller SmallChange="1" LargeChange="0" Value="0" Maximum="100"
Minimum="0" IncreaseButtonOffsetX="0" DecreaseButtonOffsetX="0"
IncreaseButtonOffsetY="0" DecreaseButtonOffsetY="0"></VScroller>
</PanelBehaviorSettings>
        </cc1:C1SuperPanel>
```

Press **F5** to run your project and then click on any date in the month view area of the C1Calendar control. Observe that it moves diagonally to the lower right corner of the page before settling into a resting state.

Setting Animation Effects in Code

Complete the following steps to set animation effects:

1. Declare the following namespace into your project:
   - Visual Basic
     ```
     Imports C1.Web.UI.Controls.C1SuperPanel
     ```
   - C#
     ```
     using C1.Web.UI.Controls.C1SuperPanel;
     ```
2. Set the duration of the animation:
   - Visual Basic
     ```
     C1SuperPanel1.PanelBehaviorSettings.AnimationDuration = 1500
     ```
   - C#
     ```
     C1SuperPanel1.PanelBehaviorSettings.AnimationDuration = 1500;
     ```
3. Select an animation transition effect:
   - Visual Basic
     ```
     C1SuperPanel1.ResizeSettings.AnimationEasing = Easing.EaseOutBounce
     ```
   - C#
     ```
     C1SuperPanel1.ResizeSettings.AnimationEasing = Easing.EaseOutBounce;
     ```

   Press **F5** to run your project and then click on any date in the month view area of the C1Calendar control. Observe that it moves diagonally to the lower right corner of the page before settling into a resting state.

Client-Side Tasks

The following topics show how to use C1SuperPanel's rich client-side object model to enable user interaction on the C1SuperPanel at run-time.

Set the Height and Width at Run Time

You can easily change the height and width on the **C1SuperPanel** by setting the **Height** and **Width** properties. In this topic you'll be able to resize the control at run time.

To set the control's **Height** and **Width** properties at run time, create a new AJAX-Enabled Web site project and complete the following steps:

1. Navigate to the Visual Studio Toolbox and double-click the **C1SuperPanel** icon to add the control to your page.
2. Click on the **Source** tab to switch to Source view, and note that the body of the page looks similar to the following:

```
<body>
    <form id="form1" runat="server">
        <asp:ScriptManager ID="ScriptManager1" runat="server" />
        <div>
        </div>
        <cc1:C1SuperPanel ID="C1SuperPanel1" runat="server">
        </cc1:C1SuperPanel>
    </form>
</body>
```

3. Add the following tags just below the `</cc1:C1Superpanel>` tag to add a button and textboxes to set the **Height** and **Width** properties.

```
<br />Height: <input id="TxtHeight" type="text" size="3"
value="200"/>  Width: <input id="TxtWidth" type="text" size="3"
value="200"/>  <input id="Button3" type="button" value="Set Size"
onclick="SetSize()"/>
```

4. At the top of the page, add the following JavaScript just above the opening `<html>` tag:

```
<script language="javascript" type="text/javascript">
function SetSize()
{
var oSuperPanel =
Sys.Application.findComponent("<%=C1SuperPanel1.ClientID%>");
oSuperPanel.set_height(parseInt(document.getElementById('TxtHeight').va
lue));
oSuperPanel.set_width(parseInt(document.getElementById('TxtWidth').valu
e));
}
</script>
```

   This function will change the **C1SuperPanel**'s **Height** and **Width** properties when the button on the page is clicked.

Run the project and observe:

1. The SuperPanel control's **Height** changes to 200 and its **Width** changes to 200.

2. Enter different values in the **Height** and **Width** textboxes and click the **Set Size** button. The control will resize once again.

Specify Target Scroll Position

The following tasks show how to specify a target to scroll to using the various targets:

- Scrolling to a location within SuperPanel
- Scrolling to a child within SuperPanel
- Scrolling to the specified offset parameters

Scrolling to a Location within SuperPanel

To scroll to an (x,y) location in the content area of the **SuperPanel** control you can call the client-side **scrollTo()** function. In this example the C1SuperPanel control scrolls to 300 pixels to the right and 200 pixels down.

To use the **scrollTo()** function to scroll to a location on the SuperPanel control, complete the following:

1. Add the **C1SuperPanel** to your page.
2. Add the following style markup between the <head> and </head> tags at the top of the document:

```
<style type="text/css">
     .elements ul
    {
        padding: 0px;
        margin: 0px;
    }
    .elements ul li {
    background-color:#DDDDDD;
    border:1px solid black;
    font-weight:bolder;
    height:100px;
    padding:50px;
    position:relative;
    text-align:center;
    width:200px;
    }
```

```
    .elements li {
     float:left;
     list-style: none none outside;
    }
    </style>
```

3. In Source view add the following markup right after the `</asp:ScriptManager>` tag to create the elements class to get the style for the elements that will appear within the **C1SuperPanel** control. The SuperPanel's **Height**, **Width**, and **ContentTemplate** properties will also be modified.

```
    <div class="elements">
                <div style="padding: 25px;">
        <cc1:C1SuperPanel ID="C1SuperPanel1" runat="server"
            UseEmbeddedVisualStyles="True" VisualStyle="ArcticFox"
Width="300px" Height="300px">
            <ContentTemplate>
            <ul class="elements" style="height: 1011px; width: 1820px;">
                <li><p>0</p><a href="#" title="" class="back">go
back</a></li><li><p>1</p><a href="#" title="" class="back">go
back</a></li><li><p>2</p><a href="#" title="" class="back">go
back</a></li><li><p>3</p><a href="#" title="" class="back">go
back</a></li><li><p>4</p><a href="#" title="" class="back">go
back</a></li><li><p>5</p><a href="#" title="" class="back">go
back</a></li><li><p>6</p><a href="#" title="" class="back">go
back</a></li><li><p>7</p><a href="#" title="" class="back">go
back</a></li><li><p>8</p><a href="#" title="" class="back">go
back</a></li><li><p>9</p><a href="#" title="" class="back">go
back</a></li><li><p>10</p><a href="#" title="" class="back">go
back</a></li><li><p>11</p><a href="#" title="" class="back">go
back</a></li><li><p>12</p><a href="#" title="" class="back">go
back</a></li><li><p>13</p><a href="#" title="" class="back">go
back</a></li><li><p>14</p><a href="#" title="" class="back">go
back</a></li><li><p>15</p><a href="#" title="" class="back">go
back</a></li><li><p>16</p><a href="#" title="" class="back">go
back</a></li><li><p>17</p><a href="#" title="" class="back">go
back</a></li><li><p>18</p><a href="#" title="" class="back">go
back</a></li><li><p>19</p><a href="#" title="" class="back">go
back</a></li><li><p>20</p><a href="#" title="" class="back">go
back</a></li><li id="t1"><p>21</p><a href="#" title="" class="back">go
back</a></li><li><p>22</p><a href="#" title="" class="back">go
back</a></li><li><p>23</p><a href="#" title="" class="back">go
back</a></li><li><p>24</p><a href="#" title="" class="back">go
back</a></li><li><p>25</p><a href="#" title="" class="back">go
back</a></li><li><p>26</p><a href="#" title="" class="back">go
back</a></li><li><p>27</p><a href="#" title="" class="back">go
back</a></li><li><p>28</p><a href="#" title="" class="back">go
back</a></li><li><p>29</p><a href="#" title="" class="back">go
back</a></li>
            </ul>
        </ContentTemplate>
<PanelBehaviorSettings Enabled="True" showrounder="True" >
<HScroller Visibility="Always"  Value="0" SmallChange="10" />
        <VScroller Visibility="Always" Value="0" SmallChange="10"  />
</PanelBehaviorSettings>
        </cc1:C1SuperPanel>
```

```
        </div>
        </div>
```

4. Add the following javascript right above the <head> tag to assign the **scrollTo** function to C1SuperPanel

```
<script language="javascript" type="text/javascript">
     function scrollTo() {
         var scroll = $find('<%=C1SuperPanel1.ClientID %>');
         scroll.get_panelBehavior().scrollTo(300, 200);
     }
     </script>
```

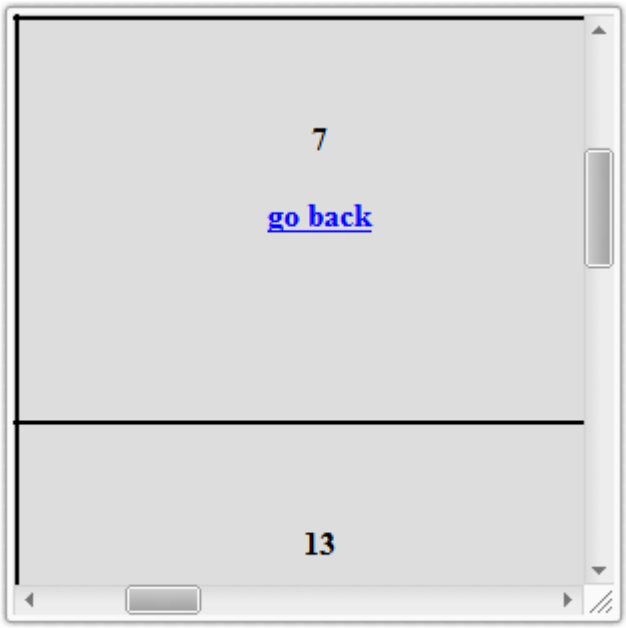
5. Create a button element and assign the **scrollTo** function to the **onclick** property.

```
        <input type="button" value="scrollTo"
onclick="scrollTo()" />   ScrollTo(300, 200)
```

6. Press **F5** to build and run your project.

Run the project and observe:

Click the **scrollTo** button to call the **scrollTo** function so it scrolls 300 pixels to the right and 200 pixels down.



Scrolling to a Child within SuperPanel

To scroll to a child in the content area of the **SuperPanel** control you can call the client-side **scrollToChild()** function.

To use the **scrollToChild()** function to scroll to a child element within C1SuperPanel, complete the following:

1. Add the **C1SuperPanel** to your page.
2. Add the following style markup between the <head> and </head> tags at the top of the document:

```
<style type="text/css">
     .elements ul
    {
        padding: 0px;
```

```
        margin: 0px;
    }
    .elements ul li {
    background-color:#DDDDDD;
    border:1px solid black;
    font-weight:bolder;
    height:100px;
    padding:50px;
    position:relative;
    text-align:center;
    width:200px;
    }

    .elements li {
     float:left;
     list-style: none none outside;
    }
    </style>
```

3. In Source view add the following markup right after the `</asp:ScriptManager>` tag to create the elements class to get the style for the elements that will appear within the **C1SuperPanel** control. The SuperPanel's **Height**, **Width**, and **ContentTemplate** properties will also be modified.

```
    <div class="elements">
                <div style="padding: 25px;">
        <cc1:C1SuperPanel ID="C1SuperPanel1" runat="server"
            UseEmbeddedVisualStyles="True" VisualStyle="ArcticFox"
Width="300px" Height="300px">
            <ContentTemplate>
            <ul class="elements" style="height: 1011px; width: 1820px;">
                <li><p>0</p><a href="#" title="" class="back">go
back</a></li><li><p>1</p><a href="#" title="" class="back">go
back</a></li><li><p>2</p><a href="#" title="" class="back">go
back</a></li><li><p>3</p><a href="#" title="" class="back">go
back</a></li><li><p>4</p><a href="#" title="" class="back">go
back</a></li><li><p>5</p><a href="#" title="" class="back">go
back</a></li><li><p>6</p><a href="#" title="" class="back">go
back</a></li><li><p>7</p><a href="#" title="" class="back">go
back</a></li><li><p>8</p><a href="#" title="" class="back">go
back</a></li><li><p>9</p><a href="#" title="" class="back">go
back</a></li><li><p>10</p><a href="#" title="" class="back">go
back</a></li><li><p>11</p><a href="#" title="" class="back">go
back</a></li><li><p>12</p><a href="#" title="" class="back">go
back</a></li><li><p>13</p><a href="#" title="" class="back">go
back</a></li><li><p>14</p><a href="#" title="" class="back">go
back</a></li><li><p>15</p><a href="#" title="" class="back">go
back</a></li><li><p>16</p><a href="#" title="" class="back">go
back</a></li><li><p>17</p><a href="#" title="" class="back">go
back</a></li><li><p>18</p><a href="#" title="" class="back">go
back</a></li><li><p>19</p><a href="#" title="" class="back">go
back</a></li><li><p>20</p><a href="#" title="" class="back">go
back</a></li><li id="t1"><p>21</p><a href="#" title="" class="back">go
back</a></li><li><p>22</p><a href="#" title="" class="back">go
back</a></li><li><p>23</p><a href="#" title="" class="back">go
back</a></li><li><p>24</p><a href="#" title="" class="back">go
back</a></li><li><p>25</p><a href="#" title="" class="back">go
```

```
back</a></li><li><p>26</p><a href="#" title="" class="back">go
back</a></li><li><p>27</p><a href="#" title="" class="back">go
back</a></li><li><p>28</p><a href="#" title="" class="back">go
back</a></li><li><p>29</p><a href="#" title="" class="back">go
back</a></li>
            </ul>
        </ContentTemplate>
<PanelBehaviorSettings Enabled="True" showrounder="True" >
<HScroller Visibility="Always"  Value="0" SmallChange="10" />
        <VScroller Visibility="Always" Value="0" SmallChange="10"  />
</PanelBehaviorSettings>
        </cc1:C1SuperPanel>
        </div>
        </div>
```

4. Add the following javascript right above the <head> tag to assign the **scrollToChild** function to C1SuperPanel

```
        function scrollToChild() {
            var scroll = $find('<%=C1SuperPanel1.ClientID %>');
            var d = $get('t1');
            d.style.border = "1px solid red";
            scroll.get_panelBehavior().scrollToChild(d);
        }
```

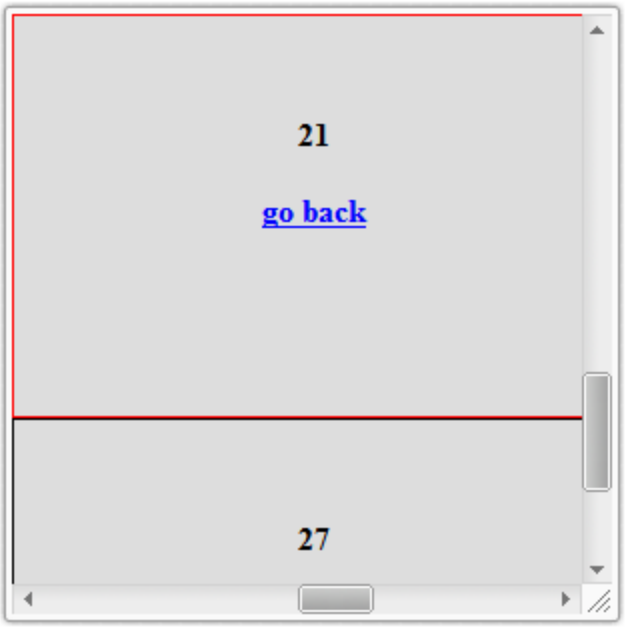
5. Create a button element and assign the **scrollToChild()** function to the **onclick** property.

```
        <input type="button" value="scrollTo"
onclick="scrollTo()" />   ScrollTo(300, 200)
```

6. Press **F5** to build and run your project.

Run the project and observe:

Click the **scrollToChild()** button to call the **scrollToChild()** function so it scrolls to the child item and adds a red border around the child element like the following:



Scrolling to the Offset

To scroll to a offset of 100 pixels in the content area of the **SuperPanel** control you can call the client-side **scrollOffSet()** function.

To use the **scrollOffSet()** function to scroll to a offset of 100 pixels in the SuperPanel control, complete the following:

1. Add the **C1SuperPanel** to your page.
2. Add the following style markup between the <head> and </head> tags at the top of the document:

```
<style type="text/css">
     .elements ul
    {
        padding: 0px;
        margin: 0px;
    }
    .elements ul li {
    background-color:#DDDDDD;
    border:1px solid black;
    font-weight:bolder;
    height:100px;
    padding:50px;
    position:relative;
    text-align:center;
    width:200px;
    }

    .elements li {
     float:left;
     list-style: none none outside;
    }
```

```
    </style>
```

3. In Source view add the following markup right after the `</asp:ScriptManager>` tag to create the elements class to get the style for the elements that will appear within the **C1SuperPanel** control. The SuperPanel's **Height**, **Width**, and **ContentTemplate** properties will also be modified.

```
    <div class="elements">
                <div style="padding: 25px;">
        <cc1:C1SuperPanel ID="C1SuperPanel1" runat="server"
            UseEmbeddedVisualStyles="True" VisualStyle="ArcticFox"
Width="300px" Height="300px">
            <ContentTemplate>
            <ul class="elements" style="height: 1011px; width: 1820px;">
                <li><p>0</p><a href="#" title="" class="back">go
back</a></li><li><p>1</p><a href="#" title="" class="back">go
back</a></li><li><p>2</p><a href="#" title="" class="back">go
back</a></li><li><p>3</p><a href="#" title="" class="back">go
back</a></li><li><p>4</p><a href="#" title="" class="back">go
back</a></li><li><p>5</p><a href="#" title="" class="back">go
back</a></li><li><p>6</p><a href="#" title="" class="back">go
back</a></li><li><p>7</p><a href="#" title="" class="back">go
back</a></li><li><p>8</p><a href="#" title="" class="back">go
back</a></li><li><p>9</p><a href="#" title="" class="back">go
back</a></li><li><p>10</p><a href="#" title="" class="back">go
back</a></li><li><p>11</p><a href="#" title="" class="back">go
back</a></li><li><p>12</p><a href="#" title="" class="back">go
back</a></li><li><p>13</p><a href="#" title="" class="back">go
back</a></li><li><p>14</p><a href="#" title="" class="back">go
back</a></li><li><p>15</p><a href="#" title="" class="back">go
back</a></li><li><p>16</p><a href="#" title="" class="back">go
back</a></li><li><p>17</p><a href="#" title="" class="back">go
back</a></li><li><p>18</p><a href="#" title="" class="back">go
back</a></li><li><p>19</p><a href="#" title="" class="back">go
back</a></li><li><p>20</p><a href="#" title="" class="back">go
back</a></li><li id="t1"><p>21</p><a href="#" title="" class="back">go
back</a></li><li><p>22</p><a href="#" title="" class="back">go
back</a></li><li><p>23</p><a href="#" title="" class="back">go
back</a></li><li><p>24</p><a href="#" title="" class="back">go
back</a></li><li><p>25</p><a href="#" title="" class="back">go
back</a></li><li><p>26</p><a href="#" title="" class="back">go
back</a></li><li><p>27</p><a href="#" title="" class="back">go
back</a></li><li><p>28</p><a href="#" title="" class="back">go
back</a></li><li><p>29</p><a href="#" title="" class="back">go
back</a></li>
            </ul>
        </ContentTemplate>
<PanelBehaviorSettings Enabled="True" showrounder="True" >
<HScroller Visibility="Always"  Value="0" SmallChange="10" />
        <VScroller Visibility="Always" Value="0" SmallChange="10"  />
</PanelBehaviorSettings>
        </cc1:C1SuperPanel>
        </div>
        </div>
```

4. Add the following javascript right above the `<head>` tag to assign the **scrollOffSet()** function to C1SuperPanel

```
function scrollOffSet() {
```

```
        var scroll = $find('<%=C1SuperPanel1.ClientID %>');
        scroll.get_panelBehavior().scrollOffset(100, 100);
    }
```

5. Create a button element and assign the **scrollOffSet()** function to the **onclick** property.

```
        <input type="button" value="scrollOffSet"
onclick="scrollOffSet()" /> Scroll a offset of 100px.
```

6. Press **F5** to build and run your project.

Run the project and observe:

Click the **scrollOffSet()** button to call the **scrollOffSet()** function so it scrolls a offset of 100px. The C1SuperPanel appears like the following: