ComponentOne

Input for ASP.NET AJAX

Table of Contents

ComponentOne Input for ASP.NET AJAX Overview

Create user-friendly, editable Web forms with **ComponentOne Input for ASP.NET AJAX**. **Input for ASP.NET AJAX** provides a collection of five input controls, including C1MaskedInput, C1DateInput, C1NumericInput, C1PercentInput, and C1CurrencyInput, to allow you to display accurate user input. Additionally, **Input for ASP.NET AJAX** includes client-side rendering technology (page 47) to increase performance in your Web forms.

Getting Started

To get started, review the following topics:

- Key Features (page 17)

- Input for ASP.NET AJAX Quick Start (page 19)

- Input for ASP.NET AJAX Samples (page 53)

The controls comprising **Input for ASP.NET AJAX** include:

- **C1CurrencyInput**

  The C1CurrencyInput control, derived from **C1NumericInput**, is specialized for editing currency values. Using the numeric editor, you can specify input without writing any custom validation logic in your application.

  For details, see Using C1CurrencyInput (page 44).

- **C1DateInput**

  The C1DateInput control, derived from **C1MaskedInput,** is specialized for editing the date and time. The **C1DateInput** control renders a date editor.

  For details, see Using C1DateInput (page 40).

- **C1MaskedInput**

  The C1MaskedInput control is the main Web control used for entering and editing information of any data type in a text form. It supports data formatting, edit mask, data validation and other features. It also supports formatted and masked editing of all data types, including additional functionality. Apart from being the main data editor control, **C1MaskedInput** also serves as the base class for specialized controls such as **C1DateInput** and **C1NumericInput**.

  For details, see Using C1MaskedInput (page 36).

- **C1NumericInput**

  The C1NumericInput control, derived from **C1MaskedInput**, is specialized for editing numeric values. Using the numeric editor, you can specify input without writing any custom validation logic in your application.

  For details, see Using C1NumericInput (page 42).

- **C1PercentInput**

The C1PercentInput control, derived from **C1NumericInput**, is specialized for editing percent values. Using the numeric editor, you can specify input without writing any custom validation logic in your application.

For details, see Using C1PercentInput (page 43).

What's New in Input for ASP.NET AJAX

No new features have been added to **Input for ASP.NET AJAX** for the 2010 v2 release of the ComponentOne Studios.

Revision History

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

What's New in Input for ASP.NET AJAX

The following new features have been added to **ComponentOne Input for ASP.NET AJAX** for 2010 v1:

- **Input for ASP.NET AJAX** now supports double byte characters.
- **Input for ASP.NET AJAX** now supports ASP.NET validation.

Class Members

The following members have been added to **SuperTooltip for WinForms** in the 2010 v1 release:

Member	Description
C1InputBase.ReadOnly Property	Gets or sets a value indicating whether the contents of the **Input for ASP.NET AJAX** control can be changed.

Installing Studio for ASP.NET AJAX

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

Studio for ASP.NET AJAX Setup Files

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

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

Samples

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

Windows XP path: C:\Documents and Settings\<username>\My Documents\ComponentOne Samples

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

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

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

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

System Requirements

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

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

Uninstalling Studio for ASP.NET AJAX

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

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

Deploying your Application in a Medium Trust Environment

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

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

ComponentOne ASP.NET AJAX 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 AJAX controls require reflection permission because LicenseManager.Validate() causes a link demand for full trust.

To add reflection permission, complete the following:

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

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

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

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

4. Save and close the web_mediumtrust.config file.

OleDbPermission

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

To add OleDbPermission, complete the following steps:

1. Open the web_mediumtrust.config file or a file created based on the web_mediumtrust.config file.
2. Add the following `<SecurityClass>` tag after the `<SecurityClasses>` tag so that it appears similar to the following:
```
<SecurityClasses>
    <SecurityClass Name="OleDbPermission"
Description="System.Data.OleDb.OleDbPermission, System.Data,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
...
</SecurityClasses>
```
3. Add the following `<IPermission>` tag after the `<NamedPermissionSets>` tag so it appears similar to the following:
```
<NamedPermissionSets>
     <PermissionSet class="NamedPermissionSet" version="1"
Name="ASP.Net">
        <IPermission class="OleDbPermission" version="1"
Unrestricted="true"/>
        ...
     </PermissionSet>
</NamedPermissionSets>
```
4. Save and close the web_mediumtrust.config file.

FileIOPermission

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

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

1. Open the web_mediumtrust.config file or a file created based on the web_mediumtrust.config file.
2. Add the following `<SecurityClass>` tag after the `<SecurityClasses>` tag so that it appears similar to the following:
```
<SecurityClasses>
    <SecurityClass Name="FileIOPermission"
Description="System.Security.Permissions.FileIOPermission, mscorlib,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
...
</SecurityClasses>
```
3. Add the following `<IPermission>` tag after the `<NamedPermissionSets>` tag so it appears similar to the following:
```
<NamedPermissionSets>
    <PermissionSet class="NamedPermissionSet" version="1"
Name="ASP.Net">
    ...
    <IPermission class="FileIOPermission" version="1"
Read="C:\SomeDir;$AppDir$" Write="$AppDir$" Append="$AppDir$"
PathDiscovery="$AppDir$" />
  ...
</PermissionSet>
</NamedPermissionSets>
```

4. Save and close the web_mediumtrust.config file.

End-User License Agreement

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

Licensing FAQs

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

What is Licensing?

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

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

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

How does Licensing Work?

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

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

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

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

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

These files are automatically added to the project.

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

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

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

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

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

Common Scenarios

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

Creating components at design time

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

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

Creating components at run time

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

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

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

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

Inheriting from licensed components

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

- Add a LicenseProvider attribute to the component.

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

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

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

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

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

Using licensed components in console applications

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

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

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

Using licensed components in Visual C++ applications

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

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

1. Build the C++ project as usual. This should create an .exe file and also a licenses.licx file with licensing information in it.
2. Copy the licenses.licx file from the app directory to the target folder (Debug or Release).
3. Copy the C1Lc.exe utility and the licensed dlls to the target folder. (Don't use the standard lc.exe, it has bugs.)
4. Use C1Lc.exe to compile the licenses.licx file. The command line should look like this:
   ```
         c1lc /target:MyApp.exe /complist:licenses.licx
   /i:C1.Win.C1FlexGrid.dll
   ```
5. Link the licenses into the project. To do this, go back to Visual Studio, right-click the project, select properties, and go to the Linker/Command Line option. Enter the following:
   ```
   /ASSEMBLYRESOURCE:Debug\MyApp.exe.licenses
   ```
6. Rebuild the executable to include the licensing information in the application.

Using licensed components with automated testing products

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

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

For example:

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

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

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

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

Troubleshooting

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

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

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

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

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

If that fails follow these steps:

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

Alternatively, follow these steps:

1. Open the project and go to the Solution Explorer window.
2. Click the **Show All Files** button on the top of the window.
3. Find the licenses.licx file and delete it.
4. Close the project and reopen it.
5. Open the main form and add an instance of each licensed control.
6. Check the Solution Explorer window, there should be a licenses.licx file there.

7. Rebuild the project using the **Rebuild All** option (not just **Rebuild**).

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

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

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

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

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

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

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

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

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

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

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

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

Option 2 – Continue to use the components you have.

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

Technical Support

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

Some methods for obtaining technical support include:

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

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

- **Peer-to-Peer Product Forums and Newsgroups**
  ComponentOne peer-to-peer product forums and newsgroups are available to exchange information, tips, and techniques regarding ComponentOne products. ComponentOne sponsors these areas as a forum for users to share information. While ComponentOne does not provide direct support in the forums and newsgroups, we periodically monitor them to ensure accuracy of information and provide comments when appropriate. Please note that a ComponentOne User Account is required to participate in the ComponentOne Product Forums.

- **Installation Issues**
  Registered users can obtain help with problems installing ComponentOne products. Contact technical support by using the online incident submission form or by phone (412.681.4738). Please note that this does not include issues related to distributing a product to end-users in an application.

- **Documentation**
  ComponentOne documentation is installed with each of our products and is also available online at HelpCentral. If you have suggestions on how we can improve our documentation, please email the Documentation team. Please note that e-mail sent to the Documentation team is for documentation feedback only. Technical Support and Sales issues should be sent directly to their respective departments.

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

Redistributable Files

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

- C1.Web.UI.2.dll
- C1.Web.UI.Controls.2.dll
- C1.Web.UI.3.dll
- C1.Web.UI.Controls.3.dll
- C1.Web.UI.4.dll
- C1.Web.UI.Controls.4.dll

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

About This Documentation

Acknowledgements

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

Firefox is a registered trademark of the Mozilla Foundation.

Safari is a registered trademark of Apple Inc.

ComponentOne

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

Corporate Headquarters

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

http://www.componentone.com

ComponentOne Doc-To-Help

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

Namespaces

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

The general namespace for ComponentOne Web products is **C1.Web**. The following code fragment shows how to declare a **C1MaskedInput** using the fully qualified name for this class:

- Visual Basic

```
Dim MaskedInput As C1.Web.UI.Controls.C1Input.C1MaskedInput
```

- C#

```
C1.Web.UI.Controls.C1Input.C1MaskedInput MaskedInput;
```

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 C1MaskedInput = C1.Web.UI.Controls.C1Input.C1MaskedInput
Imports MyMaskedInput = MyProject.Objects.C1Input.C1MaskedInput

Dim wm1 As C1MaskedInput
Dim wm2 As MyMaskedInput
```

- C#

```
using C1MaskedInput = C1.Web.UI.Controls.C1Input.C1MaskedInput;
using MyMaskedInput = MyProject.Objects.C1Input.C1MaskedInput;

 C1MaskedInput wm1;
 MyMaskedInput wm2;
```

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

Creating an AJAX-Enabled ASP.NET Project

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

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

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

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

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

The following table summarizes the installations needed:

Visual Studio Version	**Additional Installation Requirements**
Visual Studio 2008, .NET Framework 3.5	None
Visual Studio 2008 and .NET Framework 2.0 or 3.0 Visual Studio 2005 Service Pack 1	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 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.

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

Adding the Input for ASP.NET AJAX Components to a Project

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

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

Manually Adding the Studio for ASP.NET AJAX controls to the Toolbox

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

- C1CurrencyInput
- C1DateInput
- C1MaskedInput
- C1NumericInput
- C1PercentInput

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

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

Adding Studio for ASP.NET AJAX Controls to the Form

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

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

Adding a Reference to the Assembly

To add a reference to the C1.Web. UI.Controls.2 or C1.Web.UI.Controls.3 assembly:

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

The following are some of the main features of **Input for ASP.NET AJAX** that you may find useful:

- **Over 23 Built-in Masks**

  Select from over 23 built-in masks or customize your own. The C1MaskedInput control includes 14 built-in standard masks, including time and date formats, day of week chooser, and numeric range. The C1DateInput control includes 9 built-in standard masks, including short and long date and time formats. Both include custom format support.

- **Visual Alerts for Invalid Input**

  Eliminate invalid input such as alphanumeric characters in a numeric input box. You can visually alert your users with red font or display an error message. See Displaying a Message Box for an Invalid Entry (page 67) for an example of how to do this.

- **Built-in visual styles**

  Choose from five built-in visual styles to quickly change the appearance of the **Input for ASP.NET AJAX** controls. Visual styles include: **ArcticFox**, **Office2007Black**, **Office2007Blue**, **Office2007Silver**, and **Vista**. See Input for ASP.NET AJAX Visual Styles (page 45) for additional information.

- **Customize the Input for ASP.NET AJAX controls with visual styles and CSS styling**

  Easily customize the controls by pointing to your own visual styles using the **VisualStyle** property.

  Input for ASP.NET AJAX also includes CSS-supported styling so that you can use cascading style sheets to easily style the controls to match the design of your current Web site. See Adding Custom Visual Styles (page 65) for more information.

- **Supports a wide range of cultures**

  Select from an extensive range of cultures available for each **Input for ASP.NET AJAX** control. See Selecting the Culture (page 72) for details.

Input for ASP.NET AJAX Top Tips

The following tips were compiled from frequently asked user questions posted in the Studio for ASP.NET AJAX forum.

Tip 1: Add validation to your apps with visual highlighting and even masked editing.

The C1MaskedInput control uses an edit mask that prevents you from entering invalid characters in the control and provides other enhancements of the user interface. To enable masked input, just set the Mask property. For more information, see Using C1MaskedInput (page 36).

The C1MaskedInput control has a InvalidInputColor property that allows you to display a color when a user enters incorrect input. See Displaying Invalid Input Color for more information on using this property.

Tip 2: Bind a C1DateInput control to a calendar to create a date picker.

You can easily turn C1DateInput into a date picker by setting the **C1DateInput**'s WebCalendar property to the **C1Calendar** on your page. The C1DateInput control will bind to the calendar, providing a drop-down calendar from which users can choose their desired date.

Input for ASP.NET AJAX Quick Start

This section will lead you through the creation of a Web form that uses the **ComponentOne Input for ASP.NET AJAX** controls. In addition, it will show you how to change the appearance, format, and functionality of the controls. By following the steps outlined in the help, you will be able to create a rich, user-friendly Web form.

Note that for brevity, the Quick Start will show the C1MaskedInput, C1DateInput, and C1CurrencyInput controls. The C1NumericInput and C1PercentInput controls will not appear in this Quick Start since they share similar properties to the C1CurrencyInput control.

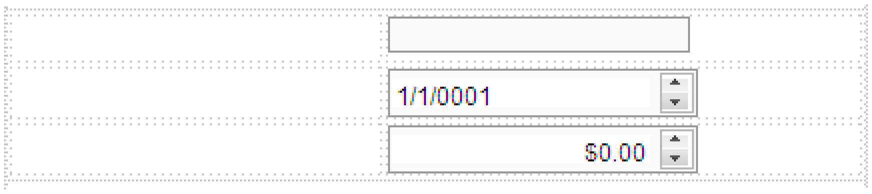
Step 1 of 5: Add Input for ASP.NET AJAX Controls to Your Form

To begin, create an ASP.NET AJAX-Enabled Web Site (page 14) and add the Input for ASP.NET AJAX (page 16) controls to your Toolbox. Note that if you are using Visual Studio 2008, you must add a **ScriptManager** control to the form. If using Visual Studio 2005, the **ScriptManager** control is automatically added to the form.

To set up your new Web form, complete the following steps:

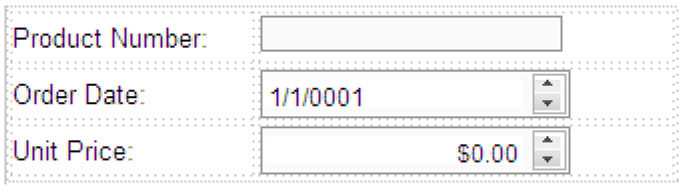
1. Click the **Design** tab located below the Document window to switch to Design view, if necessary.
2. On the page, add a table (select **Insert Table** from the **Table** menu) with two columns and three rows. The first column will be used for text and the second column for the **Input for ASP.NET AJAX** controls. The table appears on the form.
3. From the Toolbox, add the following controls to your page by completing a drag-and-drop operation placing each control in a cell in the table's second column:
   - C1MaskedInput
   - C1DateInput
   - C1CurrencyInput

   The table should look similar to the following image:



4. Add text to the table. For this example, add **Product Number:**, **Order Date:**, and **Unit Price:**, respectively. You can resize and format the table to fit your needs.

   The following image shows the table with text added:



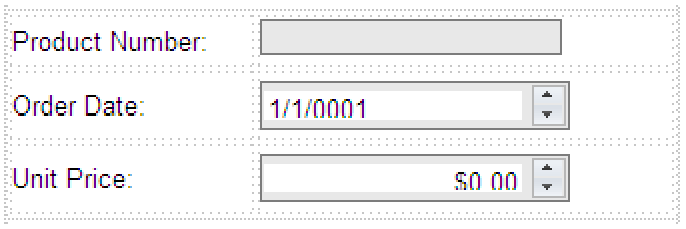
5. Switch to Source view. You can see the HTML that you created by adding the table and text in Design view.

You have successfully added the **Input for ASP.NET AJAX** controls to your Web form. The next topic shows how to change the appearance of the input boxes.

Step 2 of 5: Change the Appearance of Your Input for ASP.NET AJAX Controls

This topic shows how to change the appearance of your **Input for ASP.NET AJAX** controls using visual styles. Complete the following steps:

1. Click the **Design** tab located below the Document window to switch to Design view.
2. Select the first control, **C1MaskedInput**, and click the smart tag (▶). The **C1MaskedInput Tasks** menu appears.
3. From the Tasks menu, click the drop-down arrow next to **VisualStyle** and select **Office2007Black**.
4. Repeat steps 2 and 3 for the **C1DateInput** and **C1CurrencyInput** controls. The following image shows the appearance of the updated controls:



Product Number:	
Order Date:	1/1/0001
Unit Price:	$0.00

5. Switch to Source view. You can see the HTML that you created by changing the scheme in Design view.

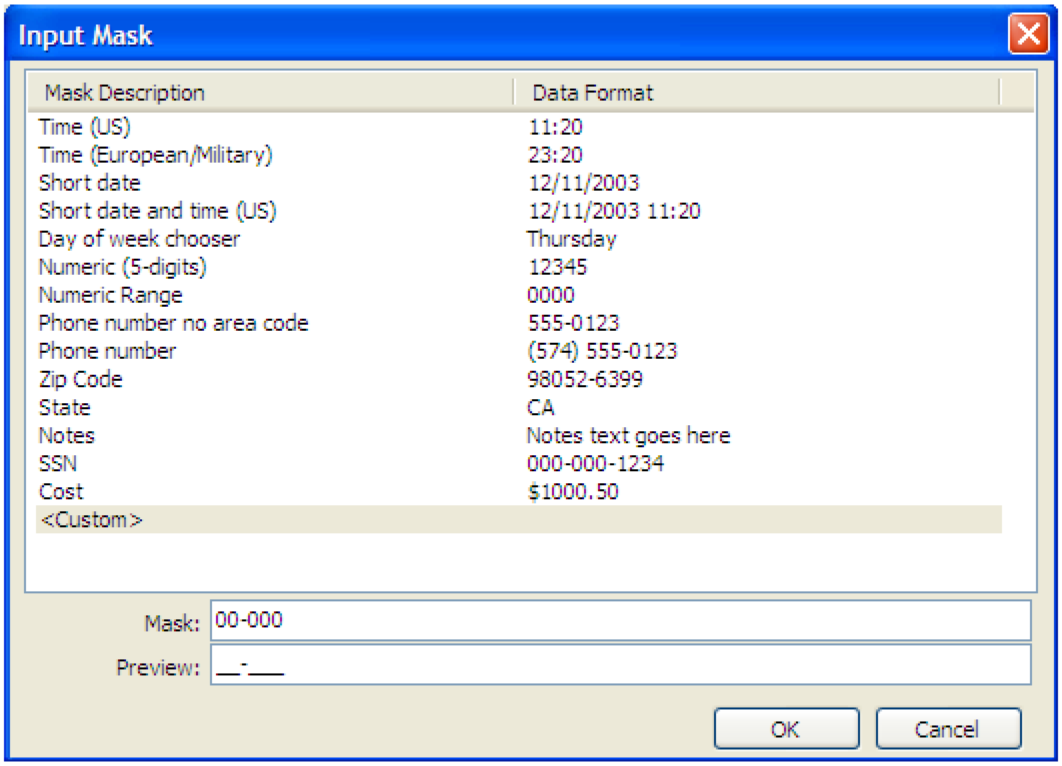
You have successfully changed the appearance of the **Input for ASP.NET AJAX** controls. The next topic shows how to format the input boxes.

Step 3 of 5: Format Your Input for ASP.NET AJAX Controls

This topic shows how to format the controls using the **Tasks** menu. To begin, click the **Design** tab located below the Document window to switch to Design view. Follow the steps below to format each of the **Input for ASP.NET AJAX** controls on your Web form.

To format the C1MaskedInput control:

1. Select the **C1MaskedInput** control and click the smart tag (▶). The **C1MaskedInput Tasks** menu appears.
2. Click the ellipsis button next to **Mask**. The **Input Mask** dialog box appears.
3. Enter **00-000** in the **Mask** text box. Note that the **Mask Description** column automatically switches to <Custom> when you start typing the mask (if the typed mask is not found in list of masks). The output value from the mask value appears in the **Preview** text box.



Mask Description	Data Format
Time (US)	11:20
Time (European/Military)	23:20
Short date	12/11/2003
Short date and time (US)	12/11/2003 11:20
Day of week chooser	Thursday
Numeric (5-digits)	12345
Numeric Range	0000
Phone number no area code	555-0123
Phone number	(574) 555-0123
Zip Code	98052-6399
State	CA
Notes	Notes text goes here
SSN	000-000-1234
Cost	$1000.50
<Custom>	

Mask: 00-000

Preview: __-___

4. Click **OK** to close the **Input Mask** dialog box.
5. In the **C1MaskedInput Tasks** menu, click the ellipsis button next to **InvalidInputColor**. The color palette appears.
6. Select red and click **OK** to close the palette.

   Note that if the user enters invalid input, such as alphanumeric characters, the input color appears red identifying the invalid entry.

To format the C1DateInput control:

1. Select the **C1DateInput** control and click the smart tag (▶). The **C1DateInput Tasks** menu appears.
2. Enter a date format in the **DateFormat** text box. Standard format characters include the following:

Preset Pattern	Name
d	Short date pattern
D	Long date pattern
t	Short time pattern
T	Long time pattern
F	Full date/time pattern(short time)
g	General date/time pattern (short time)
G	General date/time pattern (long time)
U	Universal sortable date/time pattern

The **Resulting date pattern** text box updates automatically.

3. In the **C1DateInput Tasks** menu, click the drop-down arrow next to **Date**, and select a date from the drop-down calendar.

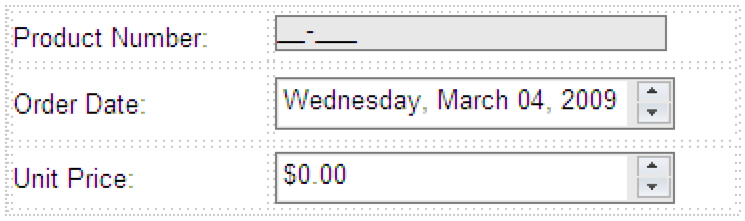
Note: You may have to resize the controls using the **Height** and **Width** properties in the Visual Studio Properties window.

To format the C1CurrencyInput control:

1. Select the **C1CurrencyInput** control and locate the TextAlign property in the Visual Studio Properties window.
2. Click the drop-down arrow and select **Left**.

You have successfully changed the format of the **Input for ASP.NET AJAX** controls. To see the HTML that you created, switch to Source view.

The following image shows the appearance of the updated controls (note that the width of each control has been set to **200px**):



Product Number:	__-___
Order Date:	Wednesday, March 04, 2009
Unit Price:	$0.00

The next topic shows how to add buttons to change the culture information for the **C1DateInput** and **C1CurrencyInput** controls.

Step 4 of 5: Add a Culture Setting

This topic demonstrates how to add code to **Button_Click** events to set the Culture for the C1DateInput and C1CurrencyInput controls. To do this, complete the following steps:

1. Click the **Design** tab located below the Document window to switch to Design view.
2. From the Toolbox, select the **Button** control and place it on your Web form (below the table) by performing a drag-and-drop operation. Repeat this step to add a second **Button** control to your Web form.
3. You should now have two **Button** controls placed next to each other on your form. Change some basic settings in the Properties window:

Button1 Properties:	**Button2 Properties:**
(ID) = FrenchBtn	**(ID)** = USEnglishBtn
Text = French Culture	**Text** = U.S. English Culture
Height = 25px	**Height** = 25px
Width = 130px	**Width** = 130px

4. Double-click the **French Culture** button to create an event handler for the button's **Click** event. Enter the following code for the **FrenchBtn_Click** event:

   - Visual Basic

```
Protected Sub FrenchBtn_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles FrenchBtn.Click
    Me.C1DateInput1.Culture = New System.Globalization.CultureInfo("fr-
FR")
    Me.C1CurrencyInput1.Culture = New
System.Globalization.CultureInfo("fr-FR")
End Sub
```

   - C#

```
protected void FrenchBtn_Click(object sender, System.EventArgs e)
{
 this.C1DateInput1.Culture = new System.Globalization.CultureInfo("fr-
FR");
 this.C1CurrencyInput1.Culture = new
System.Globalization.CultureInfo("fr-FR");
}
```

5. Double-click the **U.S. English Culture** button to create an event handler for the button's **Click** event. Enter the following code for the **USEnglishBtn_Click** event:

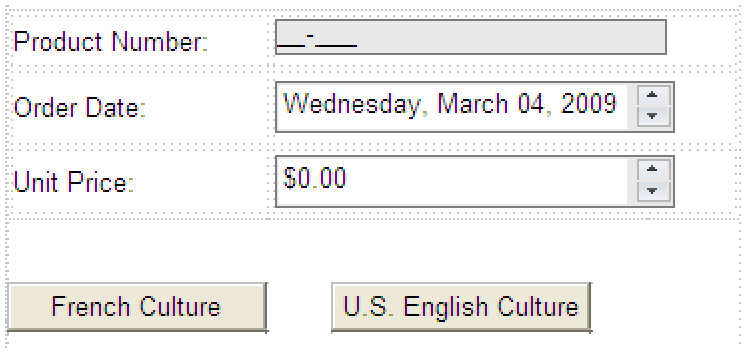
   - Visual Basic

```
Protected Sub USEnglishBtn_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles USEnglishBtn.Click
    Me.C1DateInput1.Culture = New System.Globalization.CultureInfo("en-
US")
    Me.C1CurrencyInput1.Culture = New
System.Globalization.CultureInfo("en-US")
End Sub
```

   - C#

```
protected void USEnglishBtn_Click(object sender, System.EventArgs e)
{
 this.C1DateInput1.Culture = new System.Globalization.CultureInfo("en-
US");
 this.C1CurrencyInput1.Culture = new
System.Globalization.CultureInfo("en-US");
}
```

You have successfully added two button controls with culture information to your Web form. The following image shows the appearance of the updated Web form:

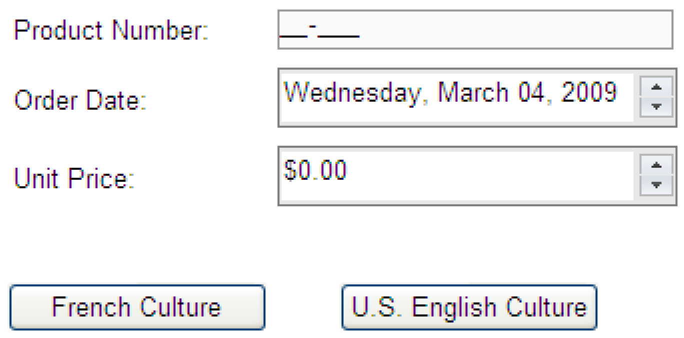


Product Number:	__-___
Order Date:	Wednesday, March 04, 2009
Unit Price:	$0.00

The next topic shows how to run the application. It also lists tasks for you to complete to observe the functionality of the Web form.

Step 5 of 5: Run Your Quick Start Web Application

Click the **Start Debugging** button to run your application. The following image shows the Quick Start Web form after completing each main step in the quick start (steps 1 – 4):



Product Number:	__-___
Order Date:	Wednesday, March 04, 2009
Unit Price:	$0.00

To observe the changes, complete the following tasks:

- Enter numeric input in the **Product Number** input box. Numeric characters are valid. Try entering an alphanumeric value (for example, *a*) and notice the input box displays red text. Also if you attempt to enter more than 5 numeric input characters (the custom mask), the input box displays red text and won't allow you to enter invalid input.
- To change the Order Date (C1DateInput control) input, complete the following tasks:
    - With your mouse pointer, click the Up/Down spin buttons.
    - Click inside the **Order Date** input box and press your keyboard UP/DOWN ARROWS.
- To change the Unit Price (C1CurrencyInput control) input, complete the following tasks:
    - With your mouse pointer, click the Up/Down spin buttons.
    - Click inside the **Unit Price** input box and press your keyboard UP/DOWN ARROWS or select the current unit price and type a new unit price.
- To change the culture to French for the C1DateInput and C1CurrencyInput controls, click the **French Culture** button.
- To change the culture back to U.S. English for the C1DateInput and C1CurrencyInput controls, click the **U.S. English Culture** button.

Congratulations!

You have successfully created a basic Web form with three different **Input for ASP.NET AJAX** controls. Additionally, you customized the controls and included culture information to increase the performance in your Web form.

Design-Time Support

Input for ASP.NET AJAX provides visual editing to make it easier to create Web input controls. The following section details each type of support available in **Input for ASP.NET AJAX**.

Invoking the Tasks Menus

In Visual Studio, each control in **Input for ASP.NET AJAX** includes a smart tag. A smart tag represents a short-cut **Tasks** menu that provides the most commonly used properties in each control. You can invoke each control's **Tasks** menu by clicking on the smart tag (▶) in the upper-right corner of the control. For more information on how to use the smart tags for each control in **Input for ASP.NET AJAX**, see Input for ASP.NET AJAX Smart Tags (page 25).

Invoking the Context Menus

You can easily configure any of the **Input for ASP.NET AJAX** components at design time by using its associated context menu. For more information on **Input for ASP.NET AJAX** context menus, see the Input for ASP.NET AJAX Context Menus (page 33).

Showing the Input for ASP.NET AJAX Control's Properties

You can access the properties for any of **Input for ASP.NET AJAX's** controls simply by right-clicking on the control and selecting **Properties** or by selecting the class from the drop-down list box of the **Properties** window.

Input for ASP.NET AJAX Smart Tags

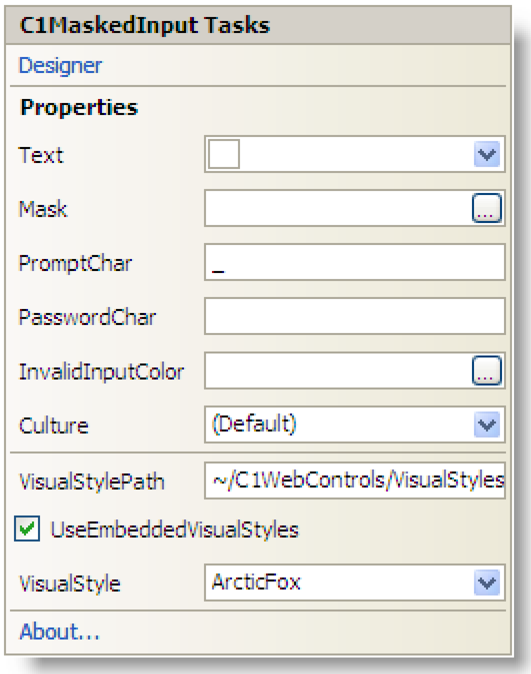
In Visual Studio, each control in **Input for ASP.NET AJAX** includes a smart tag (▶). A smart tag represents a short-cut **Tasks** menu that provides the most commonly used properties in each control.

The following topics introduce each smart tag for the **Input for ASP.NET AJAX** controls.

C1MaskedInput Smart Tag

The C1MaskedInput control provides quick and easy access to the most common C1MaskedInput properties through its smart tag.

To access the **C1MaskedInput Tasks** menu, click the smart tag (▶) in the upper-right corner of the C1MaskedInput control. This will open the **C1MaskedInput Tasks** menu.


C1MaskedInput Tasks

Designer

Properties

Text	
Mask	
PromptChar	_
PasswordChar	
InvalidInputColor	
Culture	(Default)
VisualStylePath	~/C1WebControls/VisualStyles
✓ UseEmbeddedVisualStyles	
VisualStyle	ArcticFox

About...

The **C1MaskedInput Tasks** menu operates as follows:

Designer

Clicking **Designer** opens the **C1MaskedInput Designer**. For more information on the designer, see C1MaskedInput Designer (page 34).

Properties

The most common properties of the **C1MaskedInput** control. The **C1MaskedInput Tasks** menu lists the following properties:

- **Text**

  Enter text displayed to the user in the Text box.

- **Mask**

  Click the **ellipsis** button in the Mask box, and the **Input Mask** dialog box appears. You can choose from preformatted masks or enter a custom mask.

- **PromptChar**

  Enter a prompt character, displayed in the absence of user input in the control, in the PromptChar box. The default is an underscore (_).

- **PasswordChar**

  In the PasswordChar box, for a **C1MaskedInput** control with a mask, enter a character to be substituted for the actual input characters

- **InvalidInputColor**

  Click the ellipsis button next to the InvalidInputColor property to open the color palette where you can choose the color used to identify invalid input. The default value is Empty. To set the property, you must choose a color.

- **Culture**

Click the drop-down arrow in the Culture box to select a culture. Each culture has different conventions for displaying dates, time, numbers, currency, and other information.

- **VisualStylePath**

  The **VisualStylePath** property specifies the path to the folder containing built-in visual styles by default. If you want to use a custom style, you can change the **VisualStylePath** here. See Adding Custom Visual Styles (page 65) for more information on using custom styles.

- **UseEmbeddedVisualStyles**

  The **UseEmbeddedVisualStyles** check box is checked by default so that the built-in visual styles can be used. If you want to use a custom visual style, uncheck this check box. See Adding Custom Visual Styles (page 65) for more information on using custom styles.

- **Visual Style**

  Clicking the **VisualStyle** drop-down arrow allows you to select from various built-in visual styles. See Input for ASP.NET AJAX Visual Styles (page 45) for more information.

See Using C1MaskedInput (page 36) for details.

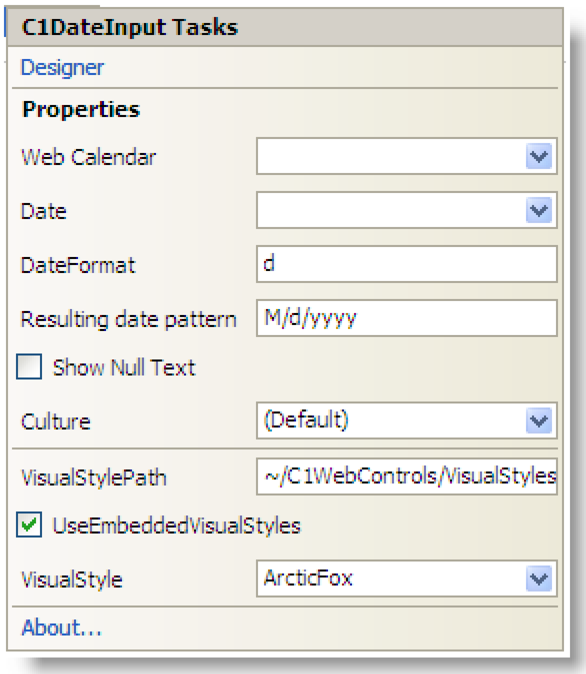
About

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

C1DateInput Smart Tag

The C1DateInput control provides quick and easy access to the most common C1DateInput properties through its smart tag.

To access the **C1DateInput Tasks** menu, click the smart tag (▶) in the upper-right corner of the C1DateInput control. This will open the **C1DateInput Tasks** menu.



The **C1DateInput Tasks** menu operates as follows:

Designer

Clicking **Designer** opens the **C1DateInput Designer**. For more information on the designer, see C1DateInput Designer (page 35).

Properties

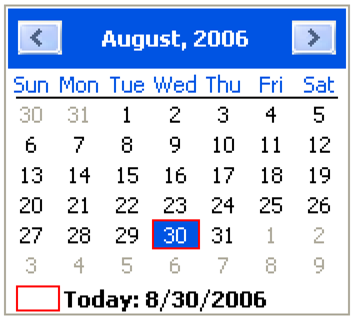
The most common properties of the **C1DateInput** control. The **C1DateInput Tasks** menu lists the following properties:

- **Web Calendar**

  Click the drop-down arrow to select the **C1Calendar** control to interact with the **C1DateInput** control. Note that the **C1Calendar** control must be added to the project for it to appear in the Web Calendar drop-down list.

- **Date**

  Enter a date in the Date box or click the drop-down arrow to select a date from the calendar:



- **DateFormat**

  Enter a date format pattern in the DateFormat box. The default value is *d*.

- **Resulting date pattern**

  Shows the resulting date format pattern, which depends on the culture (only get).

- **Show Null Text**

  Checking the **Show Null Text** check box shows null text if the **Date** value is empty and the control loses its focus. Note that the minimal date 01.01.0001 00:00:00 is used as the null date.

- **Culture**

  Click the drop-down arrow in the Culture box to select a culture. Each culture has different conventions for displaying dates, time, numbers, currency, and other information.

- **VisualStylePath**

  The **VisualStylePath** property specifies the path to the folder containing built-in visual styles by default. If you want to use a custom style, you can change the **VisualStylePath** here. See Adding Custom Visual Styles (page 65) for more information on using custom styles.

- **UseEmbeddedVisualStyles**

  The **UseEmbeddedVisualStyles** check box is checked by default so that the built-in visual styles can be used. If you want to use a custom visual style, uncheck this check box. See Adding Custom Visual Styles (page 65) for more information on using custom styles.

- **Visual Style**

  Clicking the **VisualStyle** drop-down arrow allows you to select from various built-in visual styles. See Input for ASP.NET AJAX Visual Styles (page 45) for more information.

See Using C1DateInput (page 40) for details.

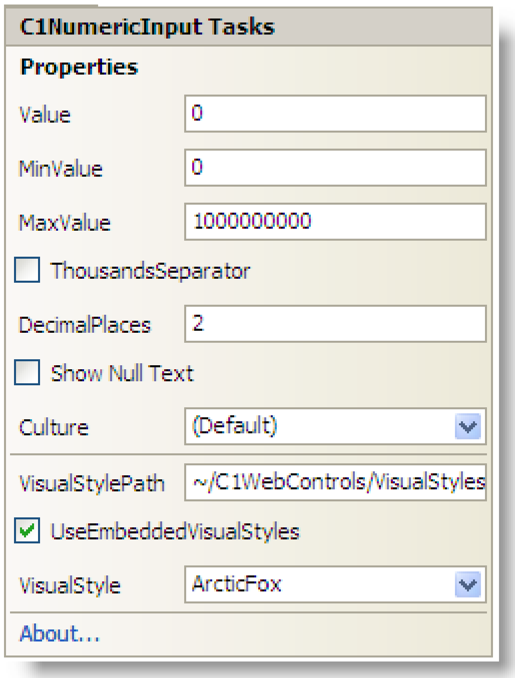
About

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

C1NumericInput Smart Tag

The C1NumericInput control provides quick and easy access to the most common C1NumericInput properties through its smart tag.

To access the **C1NumericInput Tasks** menu, click the smart tag (▶) in the upper-right corner of the C1NumericInput control. This will open the **C1NumericInput Tasks** menu.



C1NumericInput Tasks

Properties

Value	0
MinValue	0
MaxValue	1000000000
☐ ThousandsSeparator	
DecimalPlaces	2
☐ Show Null Text	
Culture	(Default)
VisualStylePath	~/C1WebControls/VisualStyles
☑ UseEmbeddedVisualStyles	
VisualStyle	ArcticFox

About...

The **C1NumericInput Tasks** menu operates as follows:

Properties

The most common properties of the **C1NumericInput** control. The **C1NumericInput Tasks** menu lists the following properties:

- **Value**

  Enter a numeric value, displayed to the user, in the Value box.

- **MinValue**

  Enter the minimum value that can be entered by the user in the MinValue box.

- **MaxValue**

Enter the maximum value that can be entered by the user in the MaxValue box.

- **ThousandsSeparator**

  Check the ThousandsSeparator check box to insert the thousands group separator between every three decimal digits (number of digits in thousands group depends on the selected Culture).

- **DecimalPlaces**

  In the DecimalPlaces box, enter the number of decimal places to display. The default value is 2.

- **Show Null Text**

  Check the **Show Null Text** check box to show NullText if the numeric **Value** is empty (equals **MinValue**) and control loses its focus.

- **Culture**

  Click the drop-down arrow in the Culture box to select a culture. Each culture has different conventions for displaying dates, time, numbers, currency, and other information.

- **VisualStylePath**

  The **VisualStylePath** property specifies the path to the folder containing built-in visual styles by default. If you want to use a custom style, you can change the **VisualStylePath** here. See Adding Custom Visual Styles (page 65) for more information on using custom styles.

- **UseEmbeddedVisualStyles**

  The **UseEmbeddedVisualStyles** check box is checked by default so that the built-in visual styles can be used. If you want to use a custom visual style, uncheck this check box. See Adding Custom Visual Styles (page 65) for more information on using custom styles.

- **Visual Style**

  Clicking the **VisualStyle** drop-down arrow allows you to select from various built-in visual styles. See Input for ASP.NET AJAX Visual Styles (page 45) for more information.

See Using C1NumericInput (page 42) for details.

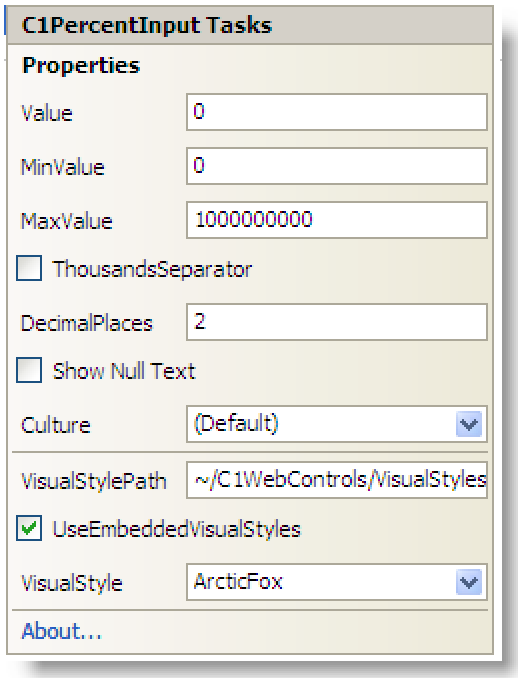
About

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

C1PercentInput Smart Tag

The C1PercentInput control provides quick and easy access to the **most common** C1PercentInput properties through its smart tag.

To access the **C1PercentInput Tasks** menu, click the smart tag (▶) in the upper-right corner of the C1PercentInput control. This will open the **C1PercentInput Tasks** menu.



Value: 0
MinValue: 0
MaxValue: 1000000000
☐ ThousandsSeparator
DecimalPlaces: 2
☐ Show Null Text
Culture: (Default)
VisualStylePath: ~/C1WebControls/VisualStyles
✓ UseEmbeddedVisualStyles
VisualStyle: ArcticFox

The **C1PercentInput Tasks** menu operates as follows:

Properties

The most common properties of the **C1PerecentEdit** control. The **C1PercentInput Tasks** menu lists the following properties:

- **Value**

  Enter a numeric value, displayed to the user, in the Value box.

- **MinValue**

  Enter the minimum value that can be entered by the user in the MinValue box.

- **MaxValue**

  Enter the maximum value that can be entered by the user in the MaxValue box.

- **ThousandsSeparator**

  Check the ThousandsSeparator check box to insert the thousands group separator between every three decimal digits (number of digits in thousands group depends on the selected Culture).

- **DecimalPlaces**

  In the DecimalPlaces box, enter the number of decimal places to display. The default value is 2.

- **Show Null Text**

  Check the **Show Null Text** check box to show NullText if the numeric **Value** is empty (equals **MinValue**) and control loses its focus.

- **Culture**

  Click the drop-down arrow in the Culture box to select a culture. Each culture has different conventions for displaying dates, time, numbers, currency, and other information.

- **VisualStylePath**

The **VisualStylePath** property specifies the path to the folder containing built-in visual styles by default. If you want to use a custom style, you can change the **VisualStylePath** here. See Adding Custom Visual Styles (page 65) for more information on using custom styles.

- **UseEmbeddedVisualStyles**

  The **UseEmbeddedVisualStyles** check box is checked by default so that the built-in visual styles can be used. If you want to use a custom visual style, uncheck this check box. See Adding Custom Visual Styles (page 65) for more information on using custom styles.

- **Visual Style**

  Clicking the **VisualStyle** drop-down arrow allows you to select from various built-in visual styles. See Input for ASP.NET AJAX Visual Styles (page 45) for more information.

See Using C1PercentInput (page 43) for details.

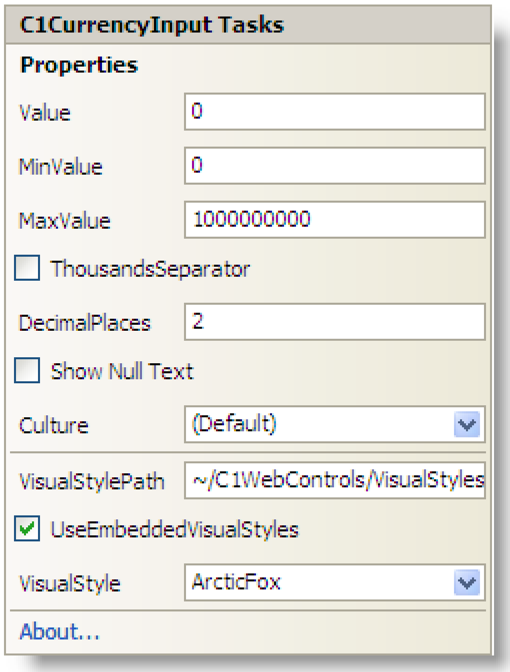
About

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

C1CurrencyInput Smart Tag

The C1CurrencyInput control provides quick and easy access to the common methods and properties through its smart tag.

To access the **C1CurrencyInput Tasks** menu, click on the smart tag (▶) in the upper-right corner of the **C1CurrencyInput** control. This will open the **C1CurrencyInput Tasks** menu.


C1CurrencyInput Tasks

Properties

Value	0
MinValue	0
MaxValue	1000000000
☐ ThousandsSeparator	
DecimalPlaces	2
☐ Show Null Text	
Culture	(Default)
VisualStylePath	~/C1WebControls/VisualStyles
✓ UseEmbeddedVisualStyles	
VisualStyle	ArcticFox

About...

The **C1CurrencyInput Tasks** menu operates as follows:

Properties

The most common properties of the **C1CurrencyInput** control. The **C1CurrencyInput Tasks** menu lists the following properties:

- **Value**

  Enter a numeric value, displayed to the user, in the Value box.

- **MinValue**

  Enter the minimum value that can be entered by the user in the MinValue box.

- **MaxValue**

  Enter the maximum value that can be entered by the user in the MaxValue box.

- **ThousandsSeparator**

  Check the ThousandsSeparator check box to insert the thousands group separator between every three decimal digits (number of digits in thousands group depends on the selected Culture).

- **DecimalPlaces**

  In the DecimalPlaces box, enter the number of decimal places to display. The default value is 2.

- **Show Null Text**

  Check the **Show Null Text** check box to show NullText if the numeric **Value** is empty (equals **MinValue**) and control loses its focus.

- **Culture**

  Click the drop-down arrow in the Culture box to select a culture. Each culture has different conventions for displaying dates, time, numbers, currency, and other information.

- **VisualStylePath**

  The **VisualStylePath** property specifies the path to the folder containing built-in visual styles by default. If you want to use a custom style, you can change the **VisualStylePath** here. See Adding Custom Visual Styles (page 65) for more information on using custom styles.

- **UseEmbeddedVisualStyles**

  The **UseEmbeddedVisualStyles** check box is checked by default so that the built-in visual styles can be used. If you want to use a custom visual style, uncheck this check box. See Adding Custom Visual Styles (page 65) for more information on using custom styles.

- **Visual Style**

  Clicking the **VisualStyle** drop-down arrow allows you to select from various built-in visual styles. See Input for ASP.NET AJAX Visual Styles (page 45) for more information.

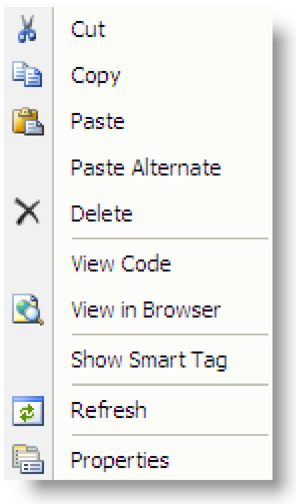
See Using C1CurrencyInput (page 44) for details.

About

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

Input for ASP.NET AJAX Context Menus

Each of the **Input for ASP.NET AJAX** controls provide a context menu for additional functionality to use at design time. Right-click on any of the **Input for ASP.NET AJAX** controls to open the following context menu:

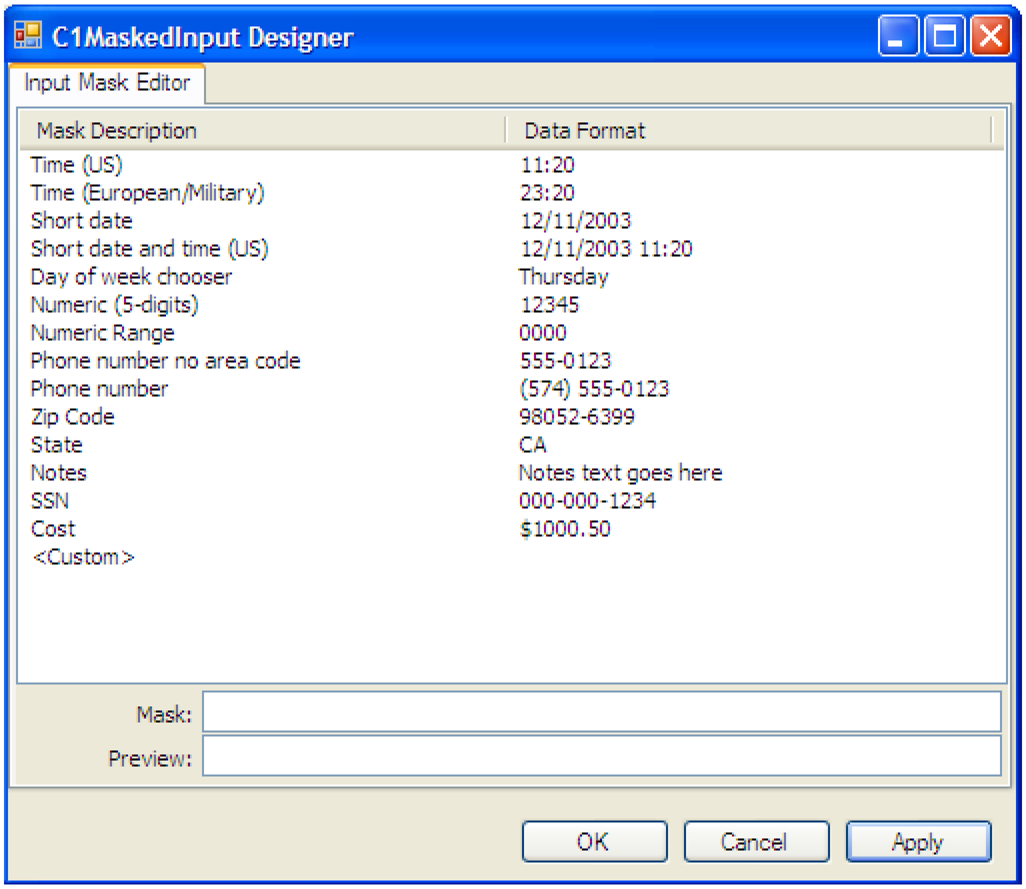


Input for ASP.NET AJAX Designers

Input for ASP.NET AJAX provides designers for C1MaskedInput and C1DateInput, allowing you to easily specify the mask or date format. These designers are described in the following topics.

C1MaskedInput Designer

To view the **C1MaskedInput Designer**, click on the smart tag (▶) in the upper-right corner of the **C1MaskedInput** control and select **Designer**. The following designer appears:



Mask Description	Data Format
Time (US)	11:20
Time (European/Military)	23:20
Short date	12/11/2003
Short date and time (US)	12/11/2003 11:20
Day of week chooser	Thursday
Numeric (5-digits)	12345
Numeric Range	0000
Phone number no area code	555-0123
Phone number	(574) 555-0123
Zip Code	98052-6399
State	CA
Notes	Notes text goes here
SSN	000-000-1234
Cost	$1000.50
<Custom>	

Input Mask Editor Tab

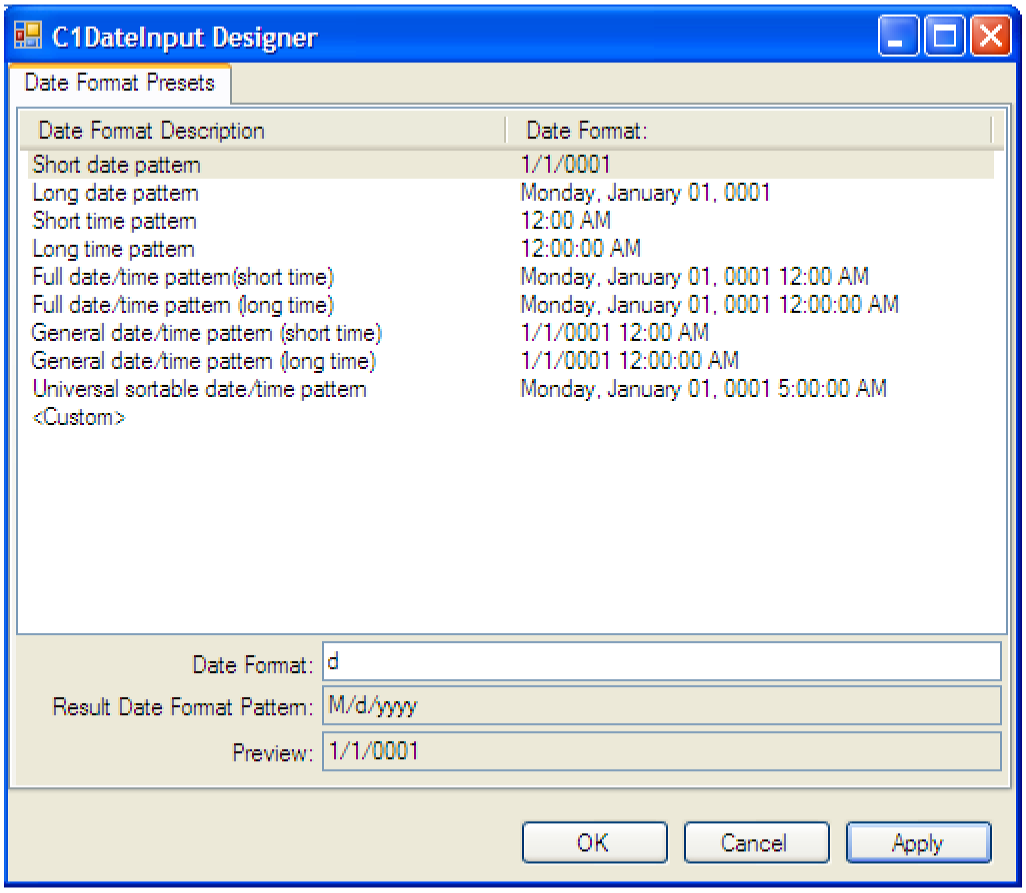
The designer's **Input Mask Editor** tab lists the C1MaskedInput control's mask options. The **Mask** text box shows the mask string composed of one or more placeholders (for example, 0, 9, #, and so on) and literals (for example, parentheses surrounding the area code of a telephone number). The **Preview** box shows you how the mask will appear in the Web browser.

Note: Not all input masks protect against non-existent values. For example, the preset 9-digit zip code mask will accept an input value of 00000-0000 even though no such zip code exists. Similarly, the preset mask for state abbreviations will allow PD to pass through even though there is no such state. Nonetheless, the preset masks are still useful as a first line defense against blatantly incorrect input.

C1DateInput Designer

To view the **C1DateInput Designer**, click on the smart tag (▶) in the upper-right corner of the **C1DateInput** control and select **Designer**.

The following designer appears:



Date Format Description	Date Format:
Short date pattern	1/1/0001
Long date pattern	Monday, January 01, 0001
Short time pattern	12:00 AM
Long time pattern	12:00:00 AM
Full date/time pattern(short time)	Monday, January 01, 0001 12:00 AM
Full date/time pattern (long time)	Monday, January 01, 0001 12:00:00 AM
General date/time pattern (short time)	1/1/0001 12:00 AM
General date/time pattern (long time)	1/1/0001 12:00:00 AM
Universal sortable date/time pattern	Monday, January 01, 0001 5:00:00 AM
<Custom>	

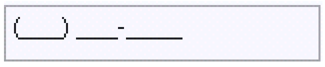
Date Format: d
Result Date Format Pattern: M/d/yyyy
Preview: 1/1/0001

Date Format Presets Tab

The designer's **Date Format Presets** tab lists the **C1DateInput** control's date format options. The **DateFormat** box shows the date format pattern composed of placeholders (dddd) and literals (for example, the divider). The **Preview** box shows you how the mask will appear in the Web browser.

Using C1MaskedInput

The **C1MaskedInput** control is basically an enhanced **TextBox** control that uses a mask to distinguish between proper and improper user input. It is the main Web control used for entering and editing information of any data type in a text form. C1MaskedInput serves as a base class for the C1DateInput and C1NumericInput controls. The following image shows a C1MaskedInput control with a phone number Mask.



Using the Mask property, you can specify the following input without writing any custom validation logic in your application:

- Mask literals (characters that should appear directly in the C1MaskedInput control); for example, the hyphen (-) in a phone number.
- The type of input required at a given position in the mask; for example, numeric or alphabetic.

- Custom input characters.

Key Benefits

The key benefits of C1MaskedInput include the following:

- It's easy to master C1MaskedInput because its most basic properties and methods are similar in behavior with the System.Windows.Forms.MaskedTextBox control at the input of text. The C1MaskedInput properties and methods are distinctive with additional functionality.
- Ability to copy and paste to and from **Input for ASP.NET AJAX** controls.
- Keyboard support:
  - LEFT/RIGHT ARROWS: move the cursor one position to the left/right.
  - HOME/END: move the cursor to the beginning or end.
  - UP/DOWN ARROWS: for enumerations and numeric ranges, increase or decrease the enumeration/numeric range value.
  - DELETE/BACKSPACE: for enumeration/numeric range, set value of enumeration/numeric range to initial value.
  - CTRL+C and CTRL+V: support for copy/paste keyboard shortcuts.
- Ability to choose a specific culture for **C1MaskedInput**, for example, English, Spanish, German, Russian, and so on.
- Ability to change most properties of **C1MaskedInput** "on-the-fly" from client script.
- Client-side events available for you to use to increase the performance of your Web form by eliminating a postback.

Defining C1MaskedInput

The C1MaskedInput control uses a mask to distinguish between proper and improper user input. You can define the mask through the visual designers, for example, the C1MaskedInput Smart Tag (page 25) or the C1MaskedInput Designer (page 34), or programmatically through the C1MaskedInput object.

For common C1MaskedInput tasks, see the C1MaskedInput Tasks (page 53) topic.

C1MaskedInput Mask Types

The following table lists some examples of masks and their behaviors:

Mask	Behavior
00/00/0000	A date (day, numeric month, year) in international date format. The "/" character is a logical date separator, and will appear to the user as the date separator appropriate to the application's current culture. Note that to specify date patterns, you can use the C1DateInput control, which provides a much richer interface for entering dates and times.
00->L<LL-0000	A date (day, month abbreviation, and year) in United States format in which the three-letter month abbreviation is displayed with an initial uppercase letter followed by two lowercase letters.
(999) 000-0000	United States phone number, area code optional. If users do not want to enter the optional characters, they can either enter spaces or place the mouse pointer directly at the position in the mask represented by the first 0.
$999,999.00	A currency value in the range of 0 to 999999. The currency, thousandth, and decimal characters will be replaced at run time with their culture-specific equivalents.

Mask is a default property for the C1MaskedInput control. If you define an edit mask, each character position in the control maps to either a special placeholder or a literal character. Literal characters, or literals, can give visual cues about the type of data being used. For example, the parentheses surrounding the area code of a telephone number and dash are literals: (412) 123-4567. The edit mask prevents you from entering invalid characters into the control and provides other enhancements of the user interface.

C1MaskedInput Characters

To enable masked input, set the Mask property to a mask string composed of one or more placeholders and literals, the following table lists available placeholders:

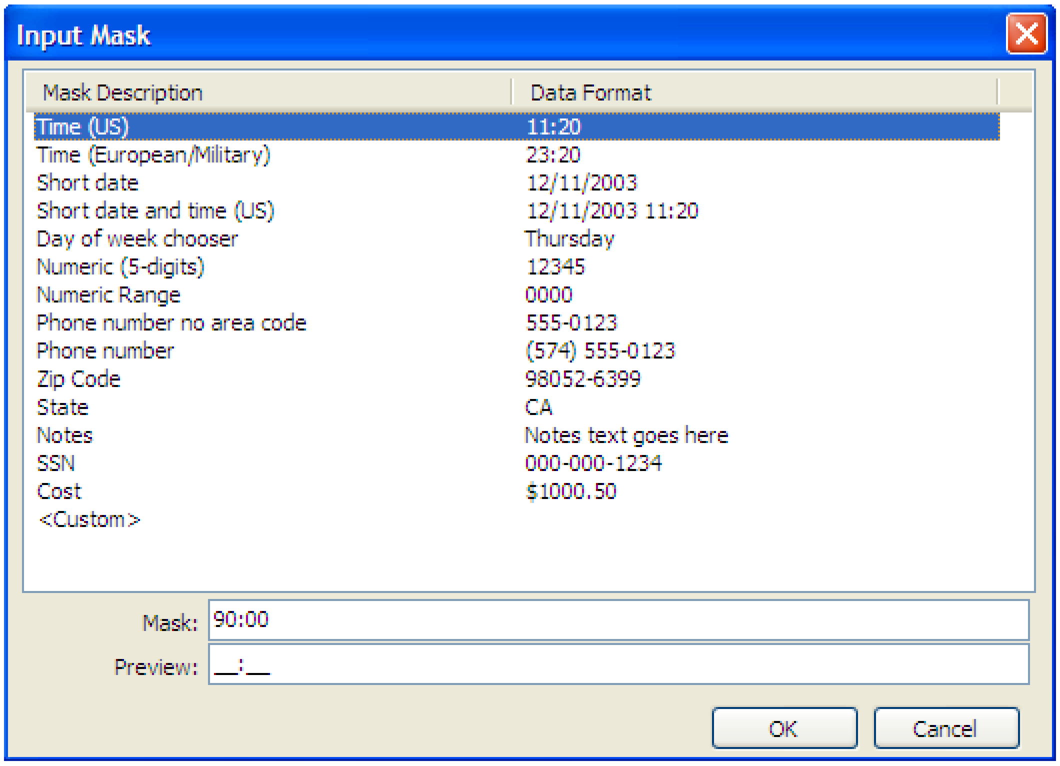
Masking Element	Description
0	Digit, required. This element will accept any single digit between 0 and 9.
9	Digit or space, optional.
#	Digit or space, optional. If this position is blank in the mask, it will be rendered as a space in the Text property. Plus (+) and minus (-) signs are allowed.
L	Letter, required. Restricts input to the ASCII letters a-z and A-Z. This mask element is equivalent to [a-zA-Z] in regular expressions.
?	Letter, optional. Restricts input to the ASCII letters a-z and A-Z. This mask element is equivalent to [a-zA-Z]? in regular expressions.
&	Character, required.
C	Character, optional. Any non-control character.
A	Alphanumeric, optional.
.	Decimal placeholder. The actual display character used will be the decimal placeholder appropriate to the Culture property.
,	Thousands placeholder. The actual display character used will be the thousands placeholder appropriate to the Culture property.
:	Time separator. The actual display character used will be the time placeholder appropriate to the Culture property.
/	Date separator. The actual display character used will be the date placeholder appropriate to the Culture property.
$	Currency symbol. The actual character displayed will be the currency symbol appropriate to the Culture property.
<	Shift down. Converts all characters that follow to lowercase.
>	Shift up. Converts all characters that follow to uppercase.
|	Disable a previous shift up or shift down.
\	Escape. Escapes a mask character, turning it into a literal. "\\" is the escape sequence for a backslash.
<<n...m**>>**	Restricts the user input to the declared numeric range, for example, <<0...255>>.
<<Value1|Value2|Value3**>>**	Restricts the user input to one of the set options. Character ("|") serves as a separator between the option values, for example, <<Option One|Option Two|Option Three>>.
All other characters	Literals. All non-mask elements will appear as themselves within the **C1MaskedInput**. Literals always occupy a static position in the mask at run time, and cannot be moved or

	deleted by the user.

If you change a mask when C1MaskedInput already contains user input filtered by a previous mask, C1MaskedInput will attempt to migrate that input into the new mask definition.

To set the C1MaskedInput.Mask property, follow these steps:

1. Select the C1MaskedInput control, and click its smart tag to open the **C1MaskedInput Tasks** menu.
2. Click the ellipsis button next to the Mask property. The **Input Mask** dialog box appears.



Mask Description	Data Format
Time (US)	11:20
Time (European/Military)	23:20
Short date	12/11/2003
Short date and time (US)	12/11/2003 11:20
Day of week chooser	Thursday
Numeric (5-digits)	12345
Numeric Range	0000
Phone number no area code	555-0123
Phone number	(574) 555-0123
Zip Code	98052-6399
State	CA
Notes	Notes text goes here
SSN	000-000-1234
Cost	$1000.50
<Custom>	

Mask: 90:00

Preview: __:__

3. Select a **Data Format** and then define the mask in the **Mask** text box. Notice the **Preview** text box displays a preview of the mask.
1. Click **OK** to close the **Input Mask** dialog box.

Using Password Characters

The C1MaskedInput control allows you to protect input text by displaying password characters. You can use the UseSystemPasswordChar property or the PasswordChar property for an empty or non-empty input control. Note that if you set the UseSystemPasswordChar property for an empty input control to **True** and specify a PasswordChar property, for example an asterisk (*), then the UseSystemPasswordChar property's default character (●) will take precedence.

For information on setting the password character, see Displaying a Password Character (page 58).

Using C1DateInput

The C1DateInput control, derived from C1MaskedInput, is specialized for editing the date and time. With the date-specific masked editing field, users can enter dates directly in the control, or use the UP/DOWN ARROW keys to increase/decrease the value of the current field. The following image shows a C1DateInput control:

Using the DateFormat property, you can specify the following input without writing any custom validation logic in your application:

- Mask literals (characters that should appear directly in the **C1DateInput** control); for example, the colon (:) in time or the separator (/) in a date.
- The type of input required at a given position in the mask; for example, numeric or alphabetic.
- Custom input characters.

Key Benefits

The key benefits of **C1DateInput** include the following:

- **C1DateInput** control renders a date editor. Use the DateFormat property to set/get the date format character or pattern.
- You can set the **C1DateInput** control to interact with the **C1WebCalendar** control. Use the WebCalendar property to integrate **C1DateInput** with **C1WebCalendar**.
- You can use the DateFormatResultPattern property to get the result of the date pattern according to the selected culture.
- Ability to choose a specific culture for **C1DateInput**, for example, English, Spanish, German, Russian, and so on. The date pattern and other aspects of date string depend on the selected Culture property.
- Client-side events available for you to use to increase the performance of your Web form by eliminating a postback.

Defining C1DateInput

You can define the date pattern through the visual designers, for example, the C1DateInput Smart Tag (page 27) or the C1DateInput Designer (page 35), or programmatically through the C1DateInput object.

When the user edits the date at run time, note the following:

- Formatted fields represented in string form, such as month or day of the week in Long date pattern, can be typed as numbers on the keyboard and their string representation is updated automatically.
- UP/DOWN ARROWS can be used to increase/decrease the current field.

C1DateInput General Properties

The following table lists general properties of the C1DateInput control:

Property	**Description**
Date	DateTime value.

DateFormat	Date format pattern or date format character (preset character).
DateFormatResultPattern	Resulting date format pattern that depends on the culture (only get).
NullText	Text that will be displayed for null date.
ShowNullText	Show null text if the Date value is empty and the control loses its focus.
WebCalendar	Gets or sets the **C1WebCalendar** control to interact with the **C1DateInput** control.

C1DateInput Format Characters

The C1DateInput format characters are case-sensitive. The following table lists standard format characters:

Preset Pattern	Name
d	Short date pattern
D	Long date pattern
t	Short time pattern
T	Long time pattern
F	Full date/time pattern(short time)
g	General date/time pattern (short time)
G	General date/time pattern (long time)
U	Universal sortable date/time pattern

C1DateInput Format Patterns

The C1DateInput patterns are case-sensitive. The following table lists standard patterns:

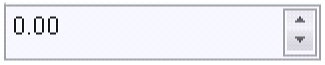
Format Pattern	Description
d	The day of the month. Single-digit days will not have a leading zero.
dd	Two-digit day of the month. Single-digit days will have a leading zero.
ddd	The abbreviated name of the day of the week.
dddd	The full name of the day of the week.
M	The numeric month. Single-digit months will not have a leading zero.
MM	The numeric month. Single-digit months will have a leading zero.
MMM	The abbreviated name of the month.
MMMM	The full name of the month.
y	The year without the century. If the year without the century is less than 10, the year is displayed with no leading zero.
yy	The year without the century. If the year without the century is less than 10, the year is displayed with a leading zero.

yyyy	Four-digit year (0000 through 9999).
h	The hour in a 12-hour clock. Single-digit hours will not have a leading zero.
hh	The hour in a 12-hour clock. Single-digit hours will have a leading zero.
H	The hour in a 24-hour clock. Single-digit hours will not have a leading zero.
HH	The hour in a 24-hour clock. Single-digit hours will have a leading zero.
m	The minute. Single-digit minutes will not have a leading zero.
mm	The minute. Single-digit minutes will have a leading zero.
s	The second. Single-digit seconds will not have a leading zero.
ss	The second. Single-digit seconds will have a leading zero.
t	The first character in the AM/PM designator.
tt	The AM/PM designator.

Note: If characters in pattern are enclosed in single quotation marks then these characters are treated as literals. For example, pattern: 'dd:' dd.MM.yyyy for date 03.07.2006 outputs string "dd: 03.07.2006".

Using C1NumericInput

The C1NumericInput control, derived from C1MaskedInput, is specialized for editing numeric values. Using the numeric editor, you can specify input without writing any custom validation logic in your application. The following image shows a C1NumericInput control:



Key Benefits

The key benefits of **C1NumericInput** include the following:

- **C1NumericInput** control renders a numeric editor.
- Ability to choose a specific culture for **C1NumericInput**, for example, English, Spanish, German, Russian, and so on. Note that **C1NumericInput** uses the selected Culture property to render number group separators (thousands separator), decimal separator, and signs.
- Numeric range support with the ability to easily change MinValue and MaxValue properties.
- Client-side events available for you to use to increase the performance of your Web form by eliminating a postback.

Defining C1NumericInput

The C1NumericInput control has numeric range support for displaying numerical data. Since the C1NumericInput control deals strictly with numbers, there is no input mask to specify. To define the C1NumericInput, you simply supply the minimum and maximum values, the number of decimal places (can be zero), and indicate whether culture-specific thousands separators should be displayed.

You can define the value of the C1NumericInput control through the C1NumericInput Smart Tag (page 29) or programmatically through the C1NumericInput object.

Note that when the user edits the value at run time, the UP ARROW or DOWN ARROW keys can be used to increase or decrease the current field.

For common C1NumericInput tasks, see the C1NumericInput Tasks (page 63) topic.

C1NumericInput General Properties

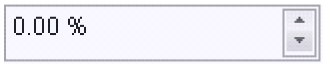
The following table lists general properties of the C1NumericInput control:

Property	**Description**
Value	Double, numeric value of the C1NumericInput control.
Text	String, displayable text according to the culture information (including group separators).
MinValue	Minimum value that can be entered.
MaxValue	Maximum value that can be entered.
DecimalPlaces	Indicates the number of decimal places to display (Default: 2).
ThousandsSeparator	Indicates whether the thousands group separator will be inserted between every three decimal digits (number of digits in thousands group depends on the selected culture).
NullText	Text that will be displayed for null value.
ShowNullText	Show null text if the Numeric value is empty and the control loses its focus.

Most of the properties and events of the C1NumericInput control are the same as the C1MaskedInput control, except for hidden properties that are not used in numeric controls (such as the following: AllowPromptAsInput, Mask, HidePromptOnLeave, PasswordChar, PromptChar, ResetOnPrompt, ResetOnSpace, ShowHintForEnumParts, SkipLiterals, UseSystemPasswordChar).

Using C1PercentInput

The C1PercentInput control, derived from C1NumericInput, is specialized for editing percent values. Using the numeric editor, you can specify input without writing any custom validation logic in your application. The following image shows a C1PercentInput control:



Key Benefits

The key benefits of C1PercentInput include the following:

- **C1PercentInput** control renders a numeric editor. C1PercentInput can be used to input percent values.
- Ability to choose a specific culture for C1PercentInput, for example, English, Spanish, German, Russian, and so on. Note that the number pattern and other aspects of number string (percent symbols and align) depends on the selected Culture property.
- Client-side events available for you to use to increase the performance of your Web form by eliminating a postback.

Note: The C1PercentInput control's properties are the same as the C1NumericInput control.

Defining C1PercentInput

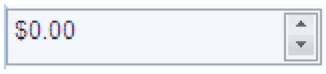
The C1PercentInput control has numeric range support for displaying numerical data. You can define the value of the C1PercentInput control through the C1PercentInput Smart Tag (page 30) or programmatically through the C1PercentInput object.

Note that when the user edits the value at run time, the UP ARROW or DOWN ARROW keys can be used to increase or decrease the current field.

Note that the C1PercentInput control's properties are the same as the C1NumericInput control. For common C1PercentInput tasks, see the C1NumericInput Tasks (page 63) topic.

Using C1CurrencyInput

The C1CurrencyInput control, derived from C1NumericInput, is specialized for editing currency values. Using the numeric editor, you can specify input without writing any custom validation logic in your application. The following image shows a C1CurrencyInput control:



Key Benefits

The key benefits of C1CurrencyInput include the following:

- C1CurrencyInput control renders a numeric editor. C1CurrencyInput can be used to input currency values.
- Ability to choose a specific culture for C1CurrencyInput, for example, English, Spanish, German, Russian, and so on. Note that the number pattern and other aspects of number string (currency symbols and align) depends on the selected Culture property.
- Client-side events available for you to use to increase the performance of your Web form by eliminating a postback.

Note: The C1CurrencyInput control's properties are the same as the C1NumericInput control.

Defining C1CurrencyInput

The C1CurrencyInput control has numeric range support for displaying numerical data. You can define the value of the C1CurrencyInput control through the C1CurrencyInput Smart Tag (page 32) or programmatically through the C1CurrencyInput object.

Note that when the user edits the value at run time, the UP ARROW or DOWN ARROW keys can be used to increase or decrease the current field.

Note that the C1CurrencyInput control's properties are the same as the C1NumericInput control. For common C1CurrencyInput tasks, see the C1NumericInput Tasks (page 63) topic.

Input for ASP.NET AJAX Appearance

Input for ASP.NET AJAX is designed to make customization easy for you. You have endless possibilities in changing the default appearance for each **Input for ASP.NET AJAX** control. **Input for ASP.NET AJAX** provides built-in Vista and Office 2007 visual styles as well as numerous property styles for the input boxes.

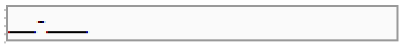
Input for ASP.NET AJAX Visual Styles

Input for ASP.NET AJAX provides five built-in visual styles for each **Input for ASP.NET AJAX** control, allowing you to automatically format the controls. The styles include the following: **ArcticFox**, **Office2007Black**, **Office2007Blue**, **Office2007Silver**, and **Vista**. For more information, see Changing the Visual Style (page 67).

The examples below show the C1MaskedInput control; however, the visual styles for all controls are the same.

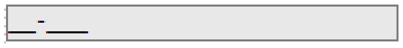
ArcticFox Style

The following image displays the **ArcticFox** style. This is the default format for all of the **Input for ASP.NET AJAX** controls:



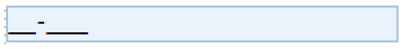
Office2007Black Style

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



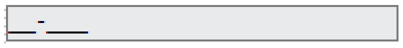
Office2007Blue Style

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



Office2007Silver Style

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



Vista Style

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

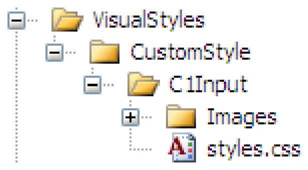
Note: An additional **Windows7** style is installed with the product, but it must be added as a custom visual style at this time. See Adding Custom Visual Styles (page 65) for more information.

Custom Visual Styles

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

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

Before adding your .css file and images, you will have to create a hierarchy of folders, the last of which will hold your files. On the top-level of your project, create a folder named "VisualStyles". Underneath the **VisualStyles** folder, create a sub-folder bearing the theme name (such as "CustomStyle), and then, beneath that, create a sub-folder named "C1Input". The image folder and .css file should be placed underneath the **C1Input** folder. The result will resemble the following:



This structure of these folders is very important; **Input for ASP.NET AJAX** will always look for the **~/VisualStyles/StyleName/C1Input/styles.css** path, as it is the default visual style path for the control.

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

For more information on customizing the appearance of **Input for ASP.NET AJAX** controls, see Adding Custom Visual Styles (page 65).

Formatting with Cascading Style Sheets

You can easily use a cascading style sheet (CSS) style to format the input boxes by assigning the **CssClass** property to your CSS class.

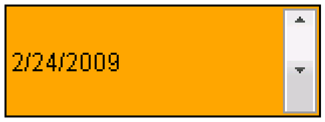
For example, click the **Source** button to view the source code for your Default.aspx page. Place the following .html source code in the <head> section of your page:

```
<head id="Head1" runat="server">
<style type="text/css">
```

```
  .ddd {height:50px;font-size:12pt;border:solid black 1px;
background:orange;}
</style>
</head>
```

This example assumes you have a C1DateInput control on your page. While still in source view, add the following code to your C1DateInput code:

```
<cc1:C1DateInput
  ID="C1DateInput1"
  runat="server"
    CssClass="ddd" Date="2009-02-24">
</cc1:C1DateInput>
```



Client-Side Functionality

ComponentOne Input for ASP.NET AJAX includes a robust client-side object model where a majority of server-side properties can be set on the client side. Client-side events can be accessed from the Visual Studio Properties window.

When an **Input for ASP.NET AJAX** control is added to a Web project, an instance of the client-side control will be created automatically. For example, if you have a C1MaskedInput control with the server-side ID of "C1MaskedEdit1", you can use the following syntax to get the client object:

```
$get("C1MaskedEdit1").control
```

OR

```
$find("C1MaskedEdit1")
```

By using **C1MaskedEdit**'s client-side functionality, you can implement many features in your Web page without having to post back to the server. Thus, using client-side methods and properties will increase the efficiency of your Web site.

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

Client-Side Properties

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

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

For more information about the client-side properties and methods, see the C1MaskedInput client-side reference section of this documentation.

Client-Side Methods

ComponentOne Input for ASP.NET AJAX includes a rich client-side object model in which several properties can be set on the client side.

The following table lists the object methods available from client-side script:

Property	Client-Side Equivalent Methods	Description
AllowPromptAsInput	set_AllowPromptAsInput get_AllowPromptAsInput	Gets or sets a value indicating whether PromptChar can be entered as valid data by the user.
BackColor	get_BackColor, set_BackColor(value)	Gets or sets the background.
BorderStyle	get_BorderStyle, set_BorderStyle(value)	Gets or sets the style of the border.
BorderWidth	get_BorderWidth, set_BorderWidth(value)	Gets or sets the width of the border.
CssClass	get_CssClass, set_CssClass(value)	CSS class name applied to the control.
Culture	set_CultureInfo, get_CultureInfo(value)	Gets or sets information about a specific culture.
DateFormat	get_DateFormat, set_DateFormat(value)	Gets or sets the date format pattern.
DecimalPlaces	get_DecimalPlaces, set_DecimalPlaces(value)	Indicates the number of decimal places to display.
ForeColor	get_ForeColor, set_ForeColor(value)	Color of the text within the control.
HideEnter	set_HideEnter, get_HideEnter(value)	If True then disable browser response on ENTER key.
HidePromptOnLeave	set_HidePromptOnLeave, get_HidePromptOnLeave(value)	Gets or sets a value indicating whether the prompt characters in the input mask are hidden when the masked text box loses focus.
Increment	get_increment(), set_increment(aIntVal)	Gets or sets how much to increase/decrease value with up/down buttons.
InvalidInputColor	set_InvalidInputColor, get_InvalidInputColor(value)	Gets or sets the color used to identify invalid input. Property omitted if the color is Empty.
Mask	set_Mask, get_Mask(value)	Gets or sets the input mask to use at run time. Mask must be a string composed of one or more of the masking elements, enumeration parts or numeric ranges.
MinValue	get_MinValue, set_MinValue(value)	Minimum value that can be entered.
MaxValue	get_MaxValue, set_MaxValue(value)	Maximum value that can

		be entered.
PasswordChar	set_PasswordChar, get_PasswordChar(value)	Gets or sets the character to be substituted for the actual input characters.
PromptChar	set_PromptChar, get_PromptChar(value)	Gets or sets the character used to represent the absence of user input in the control.
ResetOnPrompt	set_ResetOnPrompt, get_ResetOnPrompt(value)	Specifies whether to reset and skip the current position if editable, when the input character has the same value as the prompt.
ResetOnSpace	set_ResetOnSpace, get_ResetOnSpace(value)	Specifies whether to reset and skip the current position if editable, when the input is the space character.
ShowHintForEnumParts	set_ShowHintForEnumParts, get_ShowHintForEnumParts(value)	Specifies whether to show hint with list of available enumeration values for enumeration parts in mask. For example, for <<Sun|Mon|Tue>>.
SkipLiterals	set_SkipLiterals, get_SkipLiterals(value)	Specifies whether to skip current position if non-editable and the input character has the same value as the literal at that position.
Text	set_Text, get_Text(value)	Gets or sets the text as it is currently displayed to the user.
TextAlign	get_TextAlign, set_TextAlign(value)	Alignment of text in input box.
TextWithPrompts	get_TextWithPrompts(value)	Gets the text input by the user as well as any instances of the prompt character.
TextWithLiterals	get_TextWithLiterals(value)	Gets the text input by the user as well as any literal characters defined in the mask.
TextWithPromptAndLiterals	get_TextWithPromptAndLiterals(value)	Gets the text input by the user as well as any literal characters defined in the mask and any instances of the prompt character.
ThousandsSeparator	get_ThousandsSeparator, set_ThousandsSeparator(value)	Indicates whether the thousands group separator will be inserted between every three decimal digits (number of digits in thousands group

		depends on the selected culture).
ToolTip	get_ToolTip, set_ToolTip(value)	The ToolTip displayed when the mouse is over the control.
UseSystemPasswordChar	get_UseSystemPasswordChar, set_UseSystemPasswordChar	Indicates if the displayable text in the control should appear as the default password character.
Value	get_Value, set_Value(value)	The Numeric value.

The following table lists methods of the client-side object:

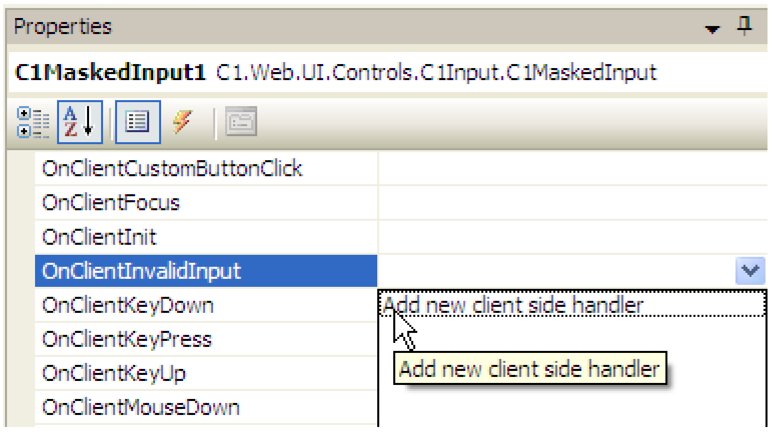
Client-Side Method	Description
clearComboItems	Clear combo items objects.
getInputElement	Returns HTML input elements
selectText(begIndex, [endIndex])	Select text in input box by index. If second parameter 'endIndex' is omitted, set caret position to new index.
setComboItems	Sets an array of combo items objects.
setComboListWidth(width)	Sets the width of the drop-down list.
setMaxComboListHeight(height)	Sets the maximum height of the drop-down list.

For information about these client-side methods and what properties can be set on the client side, see the C1MaskedInput client-side reference section of this documentation.

Client-Side Events

Input for ASP.NET AJAX includes a rich client-side object model with several client-side events.

To add a handler for a client-side event, click one of the **OnClient** properties in the Visual Studio Properties window. A drop-down list appears with a list of JavaScript functions currently defined on your Web form and the option to add a new event handler.


Properties
C1MaskedInput1 C1.Web.UI.Controls.C1Input.C1MaskedInput
OnClientCustomButtonClick
OnClientFocus
OnClientInit
OnClientInvalidInput
OnClientKeyDown
OnClientKeyPress
OnClientKeyUp
OnClientMouseDown
Add new client side handler
Add new client side handler

When you select **Add New Client Side Handler** from the drop-down list, a new JavaScript function base will automatically be generated for you and added to your Web form.

Your screen will automatically change to Code view and highlight the new function. For example, the new client-side handler for **OnClientInvalidInput** generates the following:

```
function C1MaskedInput1_OnClientInvalidInput(aC1Edit)
{
// Put you code here.
};
```

You can then write code to handle the event.

Client-side event example

Here is an example of the **OnClientKeyDown** event handler:

```
<script type="text/javascript">
// Set numeric range to 0001 on ctrl+home
// Set numeric range to 0999 on ctrl+end
function do_OnClientKeyDown(aC1Edit, aKeyCode, aEvent){
 if(aEvent.ctrlKey) {
   if(aKeyCode == 36) { // if Home key down
     aC1Edit.set_Text("0001");
     aC1Edit.selectText(3);//set caret to new position
   }
   if(aKeyCode == 35) { // if End key down
     aC1Edit.set_Text("0999");
     aC1Edit.selectText(1, 4);//select text '999'
   }
 }
};
</script>
<cc1:C1MaskedInput
  id="C1MaskedInput1"
  runat="server"
  mask="<<0...1000>>"
  onclientkeydown="do_OnClientKeyDown"
  text="0009">
</cc1:C1MaskedInput>
```

The following provides names and descriptions of **Input for ASP.NET AJAX**'s client-side events:

Event Property	Client-Side Equivalent Methods	Description	Parameter
OnClientBlur	get_OnClientBlur, set_OnClientBlur(func_name)	Occurs when the control loses the input focus.	aC1Edit
OnClientCustomButtonClick	get_OnClientCustomButtonClick, set_OnClientCustomButtonClick (func_name)	Occurs when the user clicks a custom button.	aC1Edit
OnClientDateChanged	get_OnClientDateChanged, set_OnClientDateChanged(func_name)	Occurs when the user changes the Date value.	aC1Edit
OnClientFocus	get_OnClientFocus, set_OnClientFocus(func_name)	Occurs when the control receives focus.	aC1Edit
OnClientInit	get_OnClientInit, set_OnClientInit(func_name)	Occurs when the control is initialized.	aC1Edit

OnClientInvalidInput	get_OnClientInvalidInput, set_OnClientInvalidInput(func_name)	Occurs when the user enters invalid input in the control.	aC1Edit
OnClientKeyDown	get_OnClientKeyDown, set_OnClientKeyDown(func_name)	Occurs when the user presses a key.	aC1Edit, aKeyCode, aEvent
OnClientKeyPress	get_OnClientKeyPress, set_OnClientKeyPress(func_name)	Occurs when the user presses an alphanumeric key.	aC1Edit, KeyCode, aEvent
OnClientKeyUp	get_OnClientKeyUp, set_OnClientKeyUp(func_name)	Occurs when a user releases a key.	aC1Edit, aKeyCode, aEvent
OnClientMouseDown	get_OnClientMouseDown, set_OnClientMouseDown(func_name)	Occurs when the user clicks the control with either mouse button.	aC1Edit, aEvent
OnClientMouseOut	get_OnClientMouseOut, set_OnClientMouseOut(func_name)	Occurs when the user moves the mouse pointer outside the boundaries of the control.	aC1Edit, aEvent
OnClientMouseOver	get_OnClientMouseOver, set_OnClientMouseOver(func_name)	Occurs when the user moves the mouse pointer into the control.	aC1Edit, aEvent
OnClientMouseUp	get_OnClientMouseUp, set_OnClientMouseUp(func_name)	Occurs when the user releases a mouse button while the mouse pointer is over the control.	aC1Edit, aEvent
OnClientTextChanged	get_OnClientTextChanged, set_OnClientTextChanged(func_name)	Occurs when the user changes text in the control.	aC1Edit
OnClientValueBoundsExceeded	get_OnClientValueBoundsExceeded, set_OnClientValueBoundsExceeded (func_name)	Occurs immediately after the value typed in the input box is out of Min or Max bounds.	aC1Edit

For examples showing common client-side tasks, see the Client-Side Event Tasks (page 67) topic.

Input for ASP.NET AJAX Samples

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

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

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

C# Sample

Sample	Description
VisualStyles	Displays the built-in Visual Styles including: ArticFox, Office2007Black, Office2007Blue, Office2007Silver, and Vista.
Masked Input	Displays several examples of how the C1MaskedInput control can be used.
Date Input	Displays several examples of how the C1DateInput control can be used.
Numeric Input	Displays examples of the C1NumericInput, C1PercentInput, and C1CurrencyInput controls.
Password Field	Shows how the UseSystemPasswordChar and PasswordChar properties are used to mask passwords.
Culture	Shows how different cultures can be selected to display data.
Dropdown Capability	Displays examples of how drop-down lists can be used to select date in the C1DateInput, C1MaskedInput, and C1NumericInput controls.
Calendar Integration	Shows how a calendar can be used with **Input for ASP.NET AJAX** controls to select a date.

Input for ASP.NET AJAX Task-Based Help

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

Each task-based help topic also assumes that you have created a new AJAX-enabled ASP.NET project. **ComponentOne Input for ASP.NET AJAX** requires you to create an ASP.NET AJAX-Enabled project so that Microsoft ASP.NET AJAX Extensions and a **ScriptManager** control are included in your project before the **Input for ASP.NET AJAX** controls are placed on the page. This allows you to take advantage of ASP.NET AJAX and certain features such as partial-page rendering and client-script functionality of the Microsoft AJAX Library. For additional information on this topic, see Creating an AJAX-Enabled ASP.NET Project (page 14).

C1MaskedInput Tasks

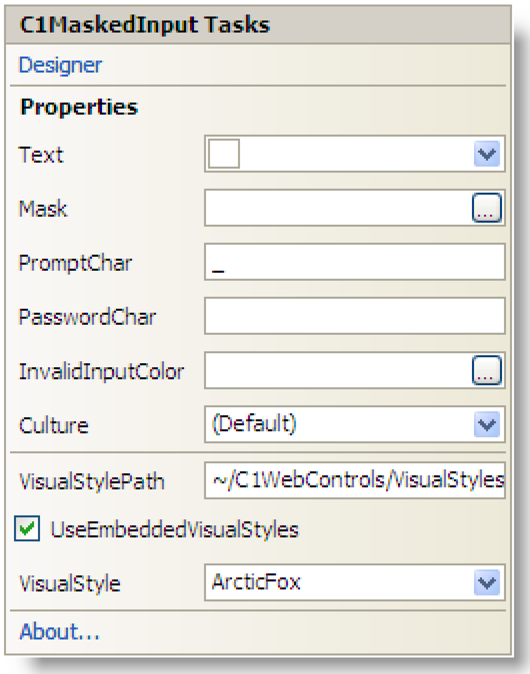
This section shows how to perform specific tasks using the C1MaskedInput control. The following topics assume that you have added a C1MaskedInput control to your Web form.

Changing the Prompt Character

At run time, the C1MaskedInput control displays the mask as a series of prompt characters (for example, # or _). The prompt characters represent each editable mask position. To change the prompt character, use the PromptChar property. This example uses the **C1MaskedInput** control with the **Phone number** mask: (999) 000-0000.

To change the prompt character using the Tasks menu:

To change the phone number PromptChar property, open the **C1MaskedInput Tasks** menu and enter the number sign (#) in the **PromptChar** text box.



Text	
Mask	
PromptChar	_
PasswordChar	
InvalidInputColor	
Culture	(Default)
VisualStylePath	~/C1WebControls/VisualStyles
✓ UseEmbeddedVisualStyles	
VisualStyle	ArcticFox

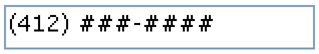
To change the prompt character using .html markup:

To change the prompt character to the number sign (#) for the **C1MaskedInput** control, use the following markup in the .aspx page:

```
<cc1:C1MaskedInput ID="C1MaskedInput1" runat="server"
      Mask="(999) 000-0000"
      Text="412"
      PromptChar="#">
</cc1:C1MaskedInput>
```

This topic illustrates the following:

Run the project and notice that the number sign (#) is displayed as the prompt character in the Web browser, as shown here:



```
(412) ###-####
```

Note that the 412 area code appears instead of the number signs since the Text property was specified for the control.

For details on hiding the prompt characters when the input box loses focus, see the Hiding the Prompt Character on Leave (page 59) topic.

Creating Day of the Week Chooser Mask

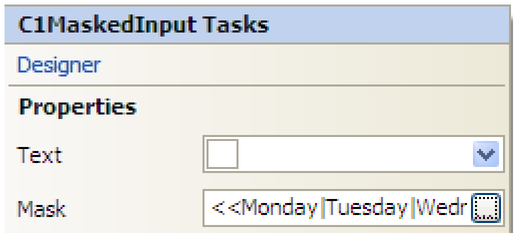
The following example demonstrates using enumeration parts in the Mask property. This example uses the C1MaskedInput control with the **Day of week chooser** mask:
<<Monday|Tuesday|Wednesday|Thursday|Friday>>.

Note that by default the **Day of week chooser** value displays Monday. You can customize this value in the designer or in the markup.

To set the day of the week using the Tasks menu:

To display Wednesday for the **Day of the week chooser** value, complete the following tasks:

1. Open the **C1MaskedInput Tasks** menu and click **Designer** to open the **C1MaskedInput Designer**.
2. Choose **Day of week chooser** for the mask value and click **OK**.
3. With the **Tasks** menu still open, type **Wednesday** in the Text text box.



Text

Mask <<Monday|Tuesday|Wedr

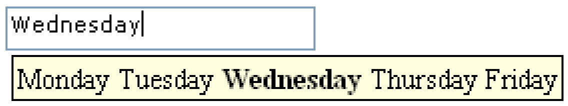
To set the day of the week using .html markup:

To display Wednesday for the **Day of the week chooser** value, use the following markup in the .aspx page:

```
<cc1:C1MaskedInput ID="C1MaskedInput1" runat="server"
Mask="&lt;&lt;Monday|Tuesday|Wednesday|Thursday|Friday&gt;&gt;"
Text="Wednesday"
</cc1:C1MaskedInput>
```

This topic illustrates the following:

Run the project, click the input box, and notice that the **Day of the week chooser** mask with Wednesday set as the text is displayed in the Web browser, as shown here:



Wednesday

Monday Tuesday Wednesday Thursday Friday

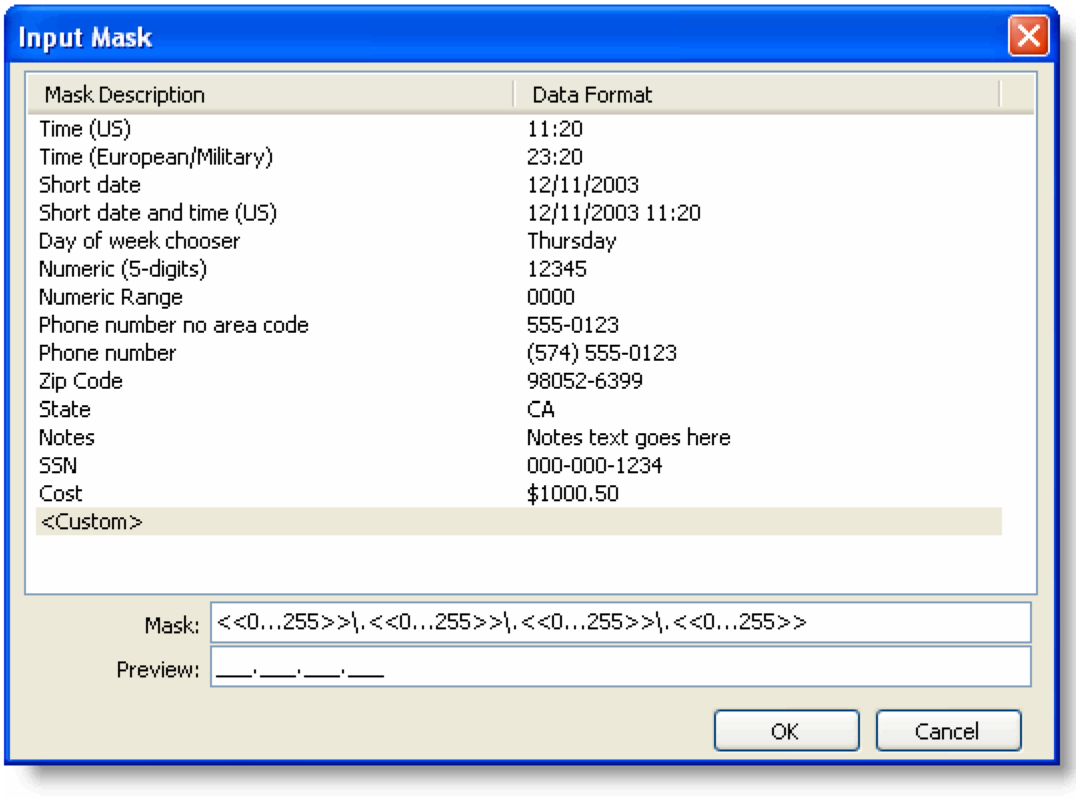
Creating an IP Address Mask

The following example demonstrates how to use numeric ranges to represent a masked text box for editing an IP address. This example uses the C1MaskedInput control with the custom mask:
<<0…255>>\.<<0…255>>\.<<0…255>>\.<<0…255>>.

To create an IP address mask using the Tasks menu:

To display the **IP address** value with specific text, complete the following tasks:

1. Open the **C1MaskedInput Tasks** menu and click the Mask property's **ellipsis** button to open the **Input Mask** dialog box.
2. Enter the following mask in the **Mask** text box: <<0...255>>\.<<0...255>>\.<<0...255>>\.<<0...255>>



Input Mask

Mask Description	Data Format
Time (US)	11:20
Time (European/Military)	23:20
Short date	12/11/2003
Short date and time (US)	12/11/2003 11:20
Day of week chooser	Thursday
Numeric (5-digits)	12345
Numeric Range	0000
Phone number no area code	555-0123
Phone number	(574) 555-0123
Zip Code	98052-6399
State	CA
Notes	Notes text goes here
SSN	000-000-1234
Cost	$1000.50
<Custom>	

Mask: <<0...255>>\.<<0...255>>\.<<0...255>>\.<<0...255>>

Preview: ___.___.___.___

Note that the Designer automatically switches to <Custom> when you start typing the mask (if the typed mask is not found in list of masks).

3. Click **OK**.
4. With the Tasks menu still open, enter **192168001001** in the Text text box.

To create an IP address mask using .html markup:

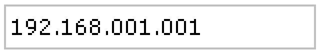
To create the masked value for an IP address, use the following markup in the .aspx page:

```
<cc1:C1MaskedInput ID="C1MaskedInput1" runat="server"
      Mask="<<0...255>>\.<<0...255>>\.<<0...255>>\.<<0...255>>"
      Text="192168001001">
</cc1:C1MaskedInput>
```

Note: One character "<" or ">" forces the next characters to shift down or shift up instructions. Character "." without "\" acts as a decimal placeholder and actual display characters used will be the decimal placeholder appropriate to the value of the Culture property.

This topic illustrates the following:

Run the project and notice that the IP address mask with the 192168001001 text is displayed in the Web browser, as shown here:


192.168.001.001

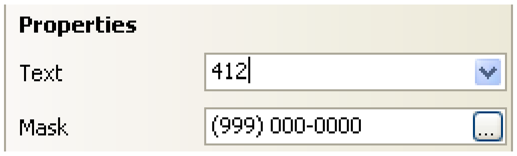
Creating a Phone Number Mask

The following example demonstrates using enumeration parts in the Mask property. This example uses the C1MaskedInput control with the **Phone Number** mask: (999) 000-0000.

To create a phone number mask using the Tasks menu:

To display the **Phone Number** value with a 412 area code, complete the following tasks:

1. Open the **C1MaskedInput Tasks** menu and click the **Mask** property's **ellipsis** button to open the **Input Mask** dialog box.
2. Choose **Phone number** for the mask value and click **OK**.
3. With the **Tasks** menu still open, type **412** in the Text text box.


Properties

Text 412

Mask (999) 000-0000

To create a phone number mask using .html markup:

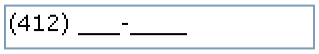
To display the **Phone Number** value with a 412 area code, use the following markup in the .aspx page:

```
<cc1:C1MaskedInput ID="C1MaskedInput1" runat="server"
      Mask="(999) 000-0000"
      Text="412"
</cc1:C1MaskedInput>
```

Note: Character "9" acts as a masking element: digit or space, optional. Character "0" acts as a masking element: digit, required. This masking element will accept any single digit between 0 and 9.

This topic illustrates the following:

Run the project and notice that the **Phone Number** mask with the 412 text is displayed in the Web browser, as shown here:


(412) ___-____

Displaying the Date Mask without Prompt Characters

To create an input box for the date that does not contain prompt characters (for example, " / /   "), use the C1MaskedInput control and set the PromptChar property to space, " ".

To create a short date mask without prompt characters using the Tasks menu:

To create a **Short Date** input box that does not contain prompt characters, complete the following tasks:

1. Open the **C1MaskedInput Tasks** menu and click the **Mask** property's **ellipsis** button to open the **Input Mask** dialog box.
2. Choose **Short Date** for the mask value and click **OK**.
3. With the **Tasks** menu still open, type a space character (" ") in the PromptChar text box. Note that you must delete the default underscore (_).

To create a short date mask without prompt characters using .html markup:

To create a **Short Date** input box that does not contain prompt characters, use the following markup in the .aspx page:

```
<cc1:C1MaskedInput ID="C1MaskedInput1" runat="server" Mask="00/00/0000"
PromptChar=" ">
</cc1:C1MaskedInput>
```

To create a short date mask without prompt characters using code:

To create a short date mask without prompt characters for the **C1MaskedInput** control, double-click the Web page to create an event handler for the **Load** event. Enter the following code for the **Page_Load** event:

- Visual Basic

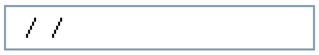
```
With C1MaskedInput1
    .Mask = "00/00/0000"
    .PromptChar = " "
End With
```

- C#

```
this.C1MaskedInput1.Mask = "00/00/0000";
this.C1MaskedInput1.PromptChar = char.Parse(" ");
```

This topic illustrates the following:

Run the project and notice that the **Short Date** mask is displayed without prompt characters, as shown here:


/ /

Displaying a Password Character

To make the text in the control display a password character, set the UseSystemPasswordChar or PasswordChar for both empty and non-empty input controls.

Option 1: Empty input control

To display a password character when the user enters text in the C1MaskedInput control:

1. Select the C1MaskedInput control and open the Visual Studio Properties window.
2. Click the drop-down arrow next to the UseSystemPasswordChar property and select **True**.

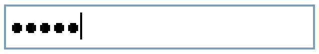
To display a password character when the user enters text in the C1MaskedInput control:

In the markup of the .aspx page insert:

```
<cc1:C1MaskedInput ID="C1MaskedInput1" runat="server"
      UseSystemPasswordChar="True">
</cc1:C1MaskedInput>
```

This topic illustrates the following:

Run the project. When you enter text in the input control, the text in the control appears as the default password character:


●●●●●

Option 2: Non-empty input control

To set a password character using the Tasks menu:

To display a password character when the user enters text in the C1MaskedInput control, complete the following steps:

1. Open the **C1MaskedInput Tasks** menu.
2. Set the Mask property for the control. For this example set it to **SSN**.
3. Enter a character, for example an asterisk (*), for the PasswordChar property.

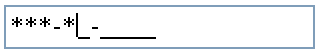
To set a password character using .html markup:

To display a password character when the user enters text in the **C1MaskedInput** control, use the following markup in the .aspx page:

```
<cc1:C1MaskedInput ID="C1MaskedInput1" runat="server"
      Mask="000-00-0000"
      PasswordChar="*">
</cc1:C1MaskedInput>
```

This topic illustrates the following:

Run the project. When you enter text in the input control, the text in the control appears as an asterisk character:


```
***-*_-____
```

Hiding the Prompt Character on Leave

You can set the HidePromptOnLeave property to **True** to hide the prompt characters when the control loses input focus.

To hide the prompt character on leave using .html markup:

In the markup of the .aspx page insert:

```
<cc1:C1MaskedInput ID="C1MaskedInput1" runat="server"
      Mask="(999) 000-0000"
      PromptChar="#"
      HidePromptOnLeave="True">
</cc1:C1MaskedInput>
```

To hide the prompt character on leave using code:

To hide the prompt character on leave for the **C1MaskedInput** control

1. Double-click the Web page to create an event handler for the **Load** event.
2. Enter the following code for the **Page_Load** event:
   - Visual Basic

```
   With C1MaskedInput1
```

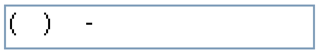
```
        .Mask = "(999) 000-0000"
        .PromptChar = "#"
        .HidePromptOnLeave = True
    End With
```

- C#

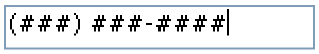
```
this.C1MaskedInput1.Mask = "(999) 000-0000";
this.C1MaskedInput1.PromptChar = char.Parse("#");
this.C1MaskedInput1.HidePromptOnLeave = true;
```

This topic illustrates the following:

Run the project. Notice that the prompt characters for the phone number mask are hidden:


( ) -

When you click inside the input box and it gets focus, the prompt characters (for this example, #) appear:


(###) ###-####

When you click outside of the input box and it loses focus, the prompt characters are hidden again. For details on changing the prompt characters, see the Changing the Prompt Character (page 54) topic.

C1DateInputTasks

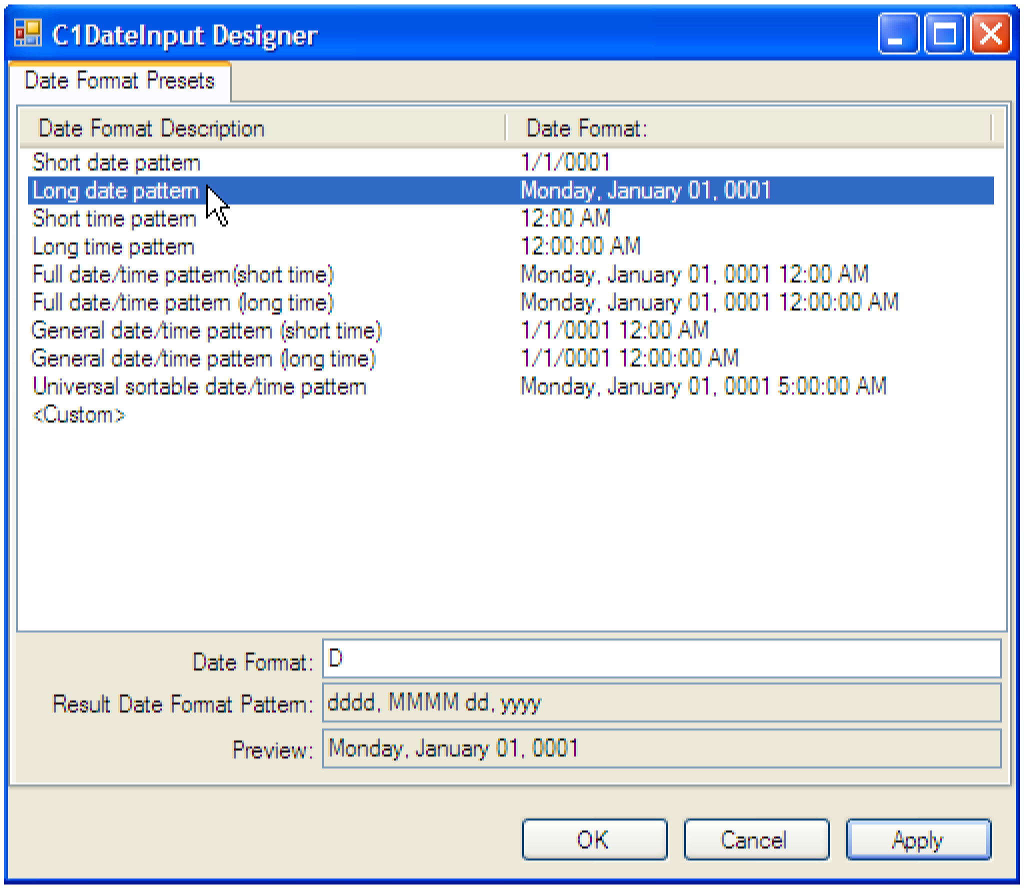
This section shows how to perform specific tasks using the C1DateInput control. The following topics assume that you have added a C1DateInput control to your Web form.

Setting the Date Format Pattern and Date

The following example demonstrates how to set the date format pattern for the **C1DateInput** control.

To set the date format pattern using the Tasks menu:

1. Open the **C1DateInput Tasks** menu and click **Designer**. The **C1DateInput Designer** appears.
2. Choose a preformatted date pattern. For this example, select **Long date pattern**.



Date Format Description	Date Format:
Short date pattern	1/1/0001
Long date pattern	Monday, January 01, 0001
Short time pattern	12:00 AM
Long time pattern	12:00:00 AM
Full date/time pattern(short time)	Monday, January 01, 0001 12:00 AM
Full date/time pattern (long time)	Monday, January 01, 0001 12:00:00 AM
General date/time pattern (short time)	1/1/0001 12:00 AM
General date/time pattern (long time)	1/1/0001 12:00:00 AM
Universal sortable date/time pattern	Monday, January 01, 0001 5:00:00 AM
<Custom>	

Date Format: D

Result Date Format Pattern: dddd, MMMM dd, yyyy

Preview: Monday, January 01, 0001

Notice that the designer shows the preview below.

3. Click **OK**.
4. With the Tasks menu still opened, click the **Date** drop-down arrow. The calendar appears.
5. Select the date selected for today's date.

To set the date format pattern using .html markup:

To display the **Long date pattern** format for the **Date Format** value, use the following markup in the .aspx page:

```
<cc1:C1DateInput ID="C1DateInput1" runat="server"
      Date="2006-12-19"
      DateFormat="D">
</cc1:C1DateInput>
```

To set the date format pattern using code:

To set the date format pattern for the C1DateInput control, double-click the Web page to create an event handler for the **Load** event. Enter the following code for the **Page_Load** event:

- Visual Basic

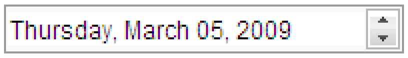
```
' Format the control as long date pattern
Me.C1DateInput1.DateFormat = "D"
' Set the date
Me.C1DateInput1.Date = "2006-12-19"
```

- C#

```
// Format the control as long date pattern
this.C1DateInput1.DateFormat = "D";
// Set the date
this.C1DateInput1.Date = DateTime.Parse("2006-12-19");
```

This topic illustrates the following:

Run the project and notice the date format pattern has been updated.

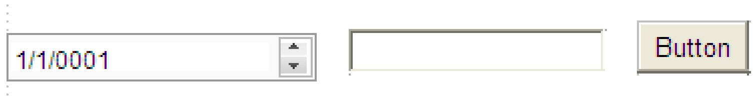


Displaying an Empty Date Value

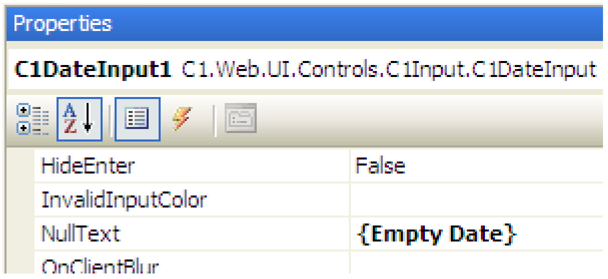
When you run your project that contains a C1DateInput control, the default date, January 1, 0001 12:00:00, automatically appears inside the control, regardless of how you have customized the controls (see Setting the Date Format Pattern and Date (page 60)). If you want the date to remain empty and not to show this default January 1, 0001 date, complete the following steps:

Using the Designer

1. Place a C1DateInput control on your form *and* two other controls, for this example use a TextBox and a Button control. These additional controls make it possible to switch from the C1DateInput control to another control on your page.



2. Switch the ShowNullText property from False to **True**.
3. Enter the desired Empty Date Value in the NullText property. For this example, use **{Empty Date}.**

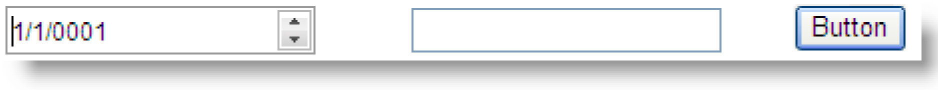


Properties

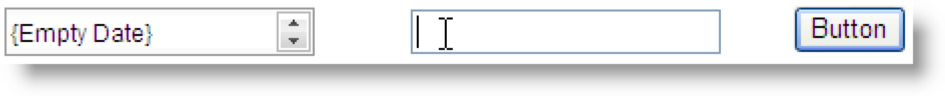
C1DateInput1 C1.Web.UI.Controls.C1Input.C1DateInput

HideEnter	False
InvalidInputColor	
NullText	**{Empty Date}**
OnClientBlur	

4. Run the project and notice the **{Empty Date}** of the C1DateInput control.
5. Select the C1DateInput so that the **{Empty Date}** is replaced with the default date.



6. Either with the cursor or by pressing the Tab button, select one of the other controls on your form.



Notice that the {Empty Date} returns to the C1DateInput control.

Note: The empty value which you have assigned to the control will appear when you first run the project. Once you select the control and toggle through the dates, the only way to have the empty value to reappear is to select another control on your page; you cannot "delete" the date within the control, you can only make the empty value text visible.

C1NumericInput Tasks

This section shows how to perform specific tasks using the C1NumericInput control.

Note that the C1CurrencyInput and C1PercentInput control's properties are the same as the C1NumericInput control; therefore, the following tasks apply to the C1CurrencyInput and C1PercentInput controls as well.

The following topics assume that you have added a C1NumericInput control to your Web form.

Enabling the Thousands Separator

The following example shows how you can easily enable the thousands separator for the C1NumericInput control.

To enable the thousands separator using the Tasks menu:

1. Open the **C1NumericInput Tasks** menu.
2. Set the **Value** of the control to **1000**.
3. Select the ThousandsSeparator checkbox in the **Tasks** menu.

To enable the thousands separator using .html markup:

To enable the ThousandsSeparator and set the **Value** to **1000**, use the following markup in the .aspx page:

```
<cc1:C1NumericInput ID="C1NumericInput1" runat="server"
      ThousandsSeparator="True"
      Value="1000">
</cc1:C1NumericInput>
```

To enable the thousands separator using code:

To enable the thousands separator for the C1NumericInput control, double-click the Web page to create an event handler for the **Load** event. Enter the following code for the **Page_Load** event:

- Visual Basic

```
' Set the numeric value
Me.C1NumericInput1.Value = 1000
' Enable the thousands separator
Me.C1NumericInput1.ThousandsSeparator = True
```

- C#

```
// Set the numeric value
this.C1NumericInput1.Value = 1000;
// Enable the thousands separator
this.C1NumericInput1.ThousandsSeparator = true;
```

Indicating the Number of Decimal Places

The following example shows how you can easily indicate the number of decimal places to display for the C1NumericInput control.

To set the decimal places value using the Tasks menu:

1. Open the **C1NumericInput Tasks** menu.
2. Set the **Value** of the control to **2.345**.
3. Enter **3** for the DecimalPlaces value.

   Note that even though you entered 2.345 for the **Value**, if you do not change the DecimalPlaces value to 3, only 2 decimal places (the default) will be displayed. That is, 2.34.

To set the decimal places value using .html markup:

To set the **Value** to **2.345** and the DecimalPlaces value to **3**, use the following markup in the .aspx page:

```
<cc1:C1NumericInput ID="C1NumericInput1" runat="server"
      DecimalPlaces="3"
      Value="2.345">
         </cc1:C1NumericInput>
```

To set the decimal places value using code:

To set the decimal places value for the C1NumericInput control, double-click the Web page to create an event handler for the **Load** event. Enter the following code for the **Page_Load** event:

- Visual Basic

```
' Set the numeric value
Me.C1NumericInput1.Value = 2.345
' Set the number of decimal places
Me.C1NumericInput1.DecimalPlaces = 3
```

- C#

```
// Set the numeric value
this.C1NumericInput1.Value = 2.345;
// Set the number of decimal places
this.C1NumericInput1.DecimalPlaces = 3;
```

Setting the Min/Max Value

The following example demonstrates the C1NumericInput control's numeric range support with the ability to easily change MinValue and MaxValue properties.

To set the numeric values using the Tasks menu:

1. Open the **C1NumericInput Tasks** menu.
2. Enter **1** for the **MinValue**.
3. Enter **1000** for the **MaxValue**.
4. With the Tasks menu still open, enter **1** in the **Value** text box.

To set the numeric values using .html markup:

To set the MinValue to **1**, the MaxValue to **1000**, and the **Value** to **1**, use the following markup in the .aspx page:

```
<cc1:C1NumericInput ID="C1NumericInput1" runat="server"
      MaxValue="1000"
      MinValue="1"
      Value="1">
</cc1:C1NumericInput>
```

To set the numeric value using code:

To set the numeric value for the **C1NumericInput** control, double-click the Web page to create an event handler for the **Load** event. Enter the following code for the **Page_Load** event:

- Visual Basic

```
With C1NumericInput1
    .MaxValue = 1000
    .MinValue = 1
    .Value = 1
End With
```

- C#

```
this.C1NumericInput1.MaxValue = 1000;
this.C1NumericInput1.MinValue = 1;
this.C1NumericInput1.Value = 1;
```

Run the project and observe the following:

- With the input control displaying 1.00, click the Down spin button with your mouse pointer and notice that the control will not display a value lower than 1.00.
- With the input control displaying 1000.00, click the Up spin button with your mouse pointer and notice that the control will not display a value higher than 1000.00.

Adding Custom Visual Styles

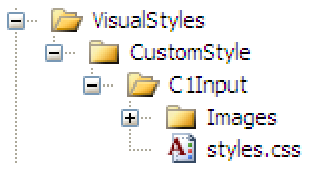
You can use the **VisualStyle**, **VisualStylePath**, and **UseEmbeddedVisualStyles** property to create a custom visual style for your **Input for ASP.NET AJAX** controls. We will use the C1MaskedInput control in this example. This topic assumes you have already added a C1MaskedInput control to your page. For more information on custom visual styles, see Custom Visual Styles (page 46).

To add a custom visual style, best practice is to copy one of the existing visual styles and customize it. In this example we will use the **C1Input ArcticFox** style.

Adding Custom Visual Styles in Design View

Complete the following steps:

1. Copy the theme folder C:\Program Files\ComponentOne\Studio for ASP.NET\C1WebUI\VisualStyles\ArcticFox\C1Input to a new folder in your Visual Studio project so the folder structure is: ~\VisualStyles\CustomStyle\C1Input.



2. Open the styles.css file in the ~/VisualStyles/CustomStyle/C1Input folder and replace any instance of "ArcticFox" with "CustomStyle". You can modify the CSS definition to customize the appearance. In this example, we'll modify the colors of the border and background of the C1MaskedInput control.
3. Locate the first instance of **.C1Input_CustomStyle** and change the border and background-color attributes so they look like the following:

```
/* =========== CustomStyle C1Input
=========================================== */
.C1Input_CustomStyle
{
      overflow: hidden;
      width: 155px;
      height: 18px;
      position: relative;
      border: #0000FF 2px solid;
      background-color: #C5B358;
      padding: 1px 0 3px 1px;
      text-align:left;
}
```

4. Save and close the styles.css file.
5. Select the C1MaskedInput control on your form and open the Visual Studio Properties window.
   a. Set the **UseEmbeddedVisualStyles** property to **False**.
   b. Make sure the **VisualStylePath** property is set to **~/VisualStyles**.
   c. Enter **CustomStyle** next to the **VisualStyle** property.

Observer that the C1MaskedInput control has adopted your custom visual style.

Adding Custom Visual Styles in Source View

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

1. Complete steps 1 through 4 under **Adding Custom Visual Styles in Design View**.
2. Click the **Source** tab to open the source view and enter `VisualStyle="CustomStyle"`, `VisualStylePath="~/VisualStyles"`, and `UseEmbeddedVisualStyles="False"` into the `<cc1:C1MaskedInput>` tag. Your XHTML will resemble the following:
   ```
   <cc1:C1MaskedInput ID="C1MaskedInput1" runat="server" Height="22px"
   VisualStyle="CustomStyle" VisualStylePath="~/VisualStyles"
   UseEmbeddedVisualStyles="False" Width="155px" />
   ```

Adding Custom Visual Styles in Code

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

1. Complete steps 1 through 4 under **Adding Custom Visual Styles in Design View**.
2. Double-click the page to place a **Page_Load** event in the code editor.
3. Set the **UseEmbeddedVisualStyles** property to **False** by adding the following code to the **Page_Load** event:
   - Visual Basic
     ```
     C1MaskedInput1.UseEmbeddedVisualStyles = False
     ```
   - C#
     ```
     C1MaskedInput1.UseEmbeddedVisualStyles = false;
     ```
4. Change the **VisualStylePath** property:
   - Visual Basic
     ```
     C1MaskedInput1.VisualStylePath = "~/VisualStyles"
     ```
   - C#
     ```
     C1MaskedInput1.VisualStylePath = "~/VisualStyles";
     ```
5. Select the custom visual style:

- Visual Basic
  ```
  C1MaskedInput1.VisualStyle = "CustomStyle"
  ```
- C#
  ```
  C1MaskedInput1.VisualStyle = "CustomStyle";
  ```

6. Run the program and observe that the C1MaskedInput control has adopted your custom visual style.

This topic illustrates the following:

In this topic, you learned how to create a custom visual style. When you build the project, your C1MaskedInput control will look like the following image:

Changing the Visual Style

You can format the **Input for ASP.NET AJAX** controls with one of five built-in visual styles. Note that you can also use your own style; see Custom Visual Styles (page 46) for more information. We will use C1MaskedInput in the following examples.

Changing the Visual Style Using the Smart Tag

You can change the style of the **Input for ASP.NET AJAX** controls at design time using the control's **Tasks** menu.

1. Click the C1MaskedInput smart tag to open the **C1MaskedInput Tasks** menu.
2. Click drop-down arrow next to **VisualStyle**.
3. Select one of the built-in styles listed. The style is applied to the C1MaskedInput control.

Changing the Visual Style in Source View

To change the visual style of the C1MaskedInput control in Source view, add `VisualStyle="Office2007Blue"` to the `<cc1:C1MaskedInput>` tag so that it resembles the following:

```
<cc1:C1MaskedInput ID="C1MaskedInput1" runat="server"
VisualStyle="Office2007Blue" Height="22px" Width="155px" />
```

Changing the Visual Style in Code

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

- Visual Basic
  ```
  C1MaskedInput1.VisualStyle = "Vista"
  ```
- C#
  ```
  C1MaskedInput1.VisualStyle = "Vista";
  ```

Client-Side Event Tasks

This section shows how to perform various client-side event tasks using the **ComponentOne Input for ASP.NET AJAX** controls.

Displaying a Message Box for an Invalid Entry

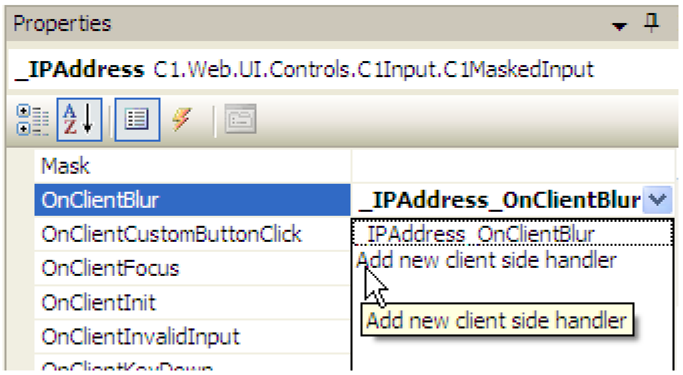
A common client-side requirement is to display a message box. The following examples show how to use the client-side event to display a message box when the user enters invalid input.

Invalid IP Address

This example uses a C1MaskedInput control with the custom mask: <<0...255>>\.<<0...255>>\.<<0...255>>\.<<0...255>>. For details on setting the control's Mask property, see Creating an IP Address Mask (page 55).

To display an alert box for an incomplete IP address, complete the following steps:

1. Select the **Input for ASP.NET AJAX** control, in this case, C1MaskedInput. Note that for this example the **ID** property for the control was changed to **_IPAddress**.
2. From the Visual Studio Properties window, locate the client-side event, OnClientBlur.
3. Click the **OnClientBlur** drop-down arrow and select **Add New Client Side Handler**.


Properties
_IPAddress C1.Web.UI.Controls.C1Input.C1MaskedInput
Mask
OnClientBlur
_IPAddress_OnClientBlur
OnClientCustomButtonClick
_IPAddress_OnClientBlur
Add new client side handler
OnClientFocus
OnClientInit
Add new client side handler
OnClientInvalidInput

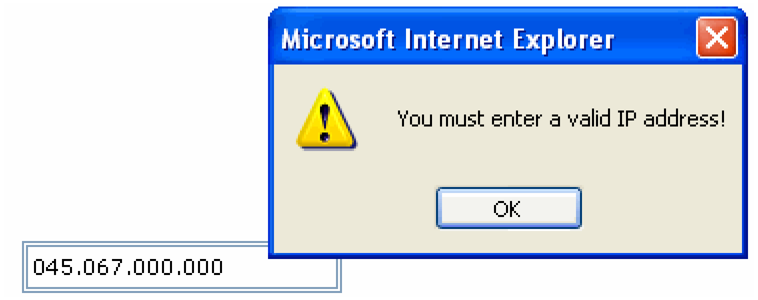
   Note that the drop-down list also lists functions currently defined on your Web form.

4. Your screen will automatically change to Code view and highlight the new function. Enter the following code inside the **OnClientBlur** function (replacing the code comment which is automatically inserted):

```
var aTextEntered = aC1Edit.get_TextWithLiterals();
//The mask is:
//<<0...255>>\.<<0...255>>\.<<0...255>>\.<<0...255>>
var arr = aTextEntered.split(".");
arr[0] = parseFloat(arr[0]);
arr[1] = parseFloat(arr[1]);
arr[2] = parseFloat(arr[2]);
arr[3] = parseFloat(arr[3]);
if(arr[0] < 1 || arr[1] < 1 || arr[2] < 1)
alert("You must enter a valid IP address!");
```

This topic illustrates the following:

Run the project. Begin typing the IP address and leave the input control before entering the entire address, and you will receive an alert box:


Microsoft Internet Explorer

You must enter a valid IP address!

Invalid Phone Number

This example uses a C1MaskedInput control with the **Phone number** mask: (999) 000-0000. If the user enters anything less than the required 10-digit phone number, an alert box will appear.

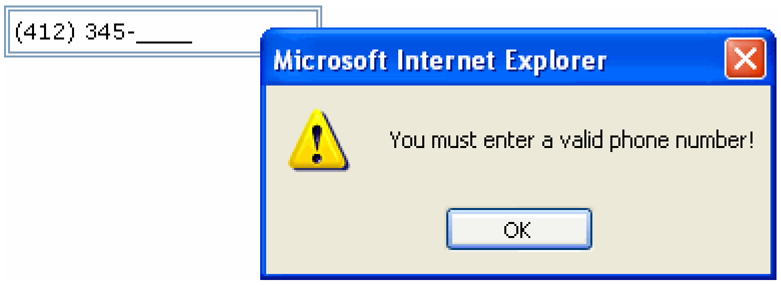
To display an alert box for an incomplete phone number, complete the following steps:

1. Select the **Input for ASP.NET AJAX** control, in this case, C1MaskedInput. Note that for this example the **ID** property for the control was changed to **_PhoneNumber**.
2. From the Properties window, locate the client-side event, **OnClientBlur**.
3. Click the **OnClientBlur** drop-down arrow and select **Add New Client Side Handler**.
4. Your screen will automatically change to Source view and highlight the new function. Enter the following code inside the **OnClientBlur** function (replacing the code comment which is automatically inserted):

```
// Get text without prompts and literals:
var sText = aC1Edit.get_Text();

// Remove all spaces from text:
sText = sText.replace(' ','');

// Full phone number must be 10 characters length:
if(sText.length < 10)
alert("You must enter a valid phone number!");
```

This topic illustrates the following:

Run the project. Begin typing the phone number and leave the input control before entering the entire 10-digit number, and you will receive an alert box:


Microsoft Internet Explorer

You must enter a valid phone number!

Rendering a Combo List

The following example renders a combo list that appears when the user presses the custom button. This example uses the C1MaskedInput control.

Add a custom button:

To implement a custom button for the C1MaskedInput control, complete the following steps:

1. Select the C1MaskedInput control.
2. From the Properties window, set the CustomButtonAlign property to **Right**.

Add the client-side function for Combo List initialization:

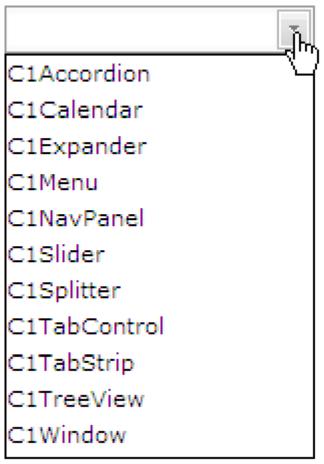
To add an event handler to render a combo list when the custom button is pressed, complete the following steps:

1. Select the C1MaskedInput control.
2. From the Properties window, locate the client-side event, **OnClientInit**.
3. Click the **OnClientInit** drop-down arrow and select **Add New Client Side Handler**.
4. Your screen will automatically change to Source view and highlight the new function. Enter code so the **OnClientInit** function looks like the following (you will be replacing the code comment which is automatically inserted):

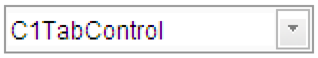
```
<script type="text/javascript" id="ComponentOneClientScript">
        function C1MaskedInput1_OnClientInit(aC1Edit) {

    var aComboItems = new Array();

    aComboItems.push({ text: "C1Accordion", value: "C1Accordion" });
    aComboItems.push({ text: "C1Calendar", value: "C1Calendar" });
    aComboItems.push({ text: "C1Expander", value: "C1Expander" });
    aComboItems.push({ text: "C1Menu", value: "C1Menu" });
    aComboItems.push({ text: "C1NavPanel", value: "C1NavPanel" });
    aComboItems.push({ text: "C1Slider", value: "C1Slider" });
    aComboItems.push({ text: "C1Splitter", value: "C1Splitter" });
    aComboItems.push({ text: "C1TabControl", value: "C1TabControl" });
    aComboItems.push({ text: "C1TabStrip", value: "C1TabStrip" });
    aComboItems.push({ text: "C1TreeView", value: "C1TreeView" });
    aComboItems.push({ text: "C1Window", value: "C1Window" });
    aC1Edit.setComboItems(aComboItems);

};
</script>
```

This topic illustrates the following:

Run the project. Click the custom button and the list of combo items appears:



Select an item from the list for it to appear in the input box:



Note that for numeric controls, the type of combo value must be JavaScript float.

For the C1DateInput, control the value must be JavaScript Date type.

For example, combo items for C1DateInput can be initialized as follow:

```
<script type="text/javascript">
function C1DateInput1_OnClientInit(aC1Edit){
   var aComboItems = new Array();
   aComboItems.push( {text:"1980/4/8", value:new Date(1980,3,8)} );
   aComboItems.push( {text:"2007/12/25", value:new Date(2007,11,25)} );
   aComboItems.push( {text:"today", value:new Date()} );
   aC1Edit.setComboItems(aComboItems);
};
</script>
</head>
<body>
   <cc1:C1DateInput ID="C1DateInput1" runat="server"
   OnClientInit="C1DateInput1_OnClientInit" CustomButtonAlign="Right">
   </cc1:C1DateInput>
```

Using HTML text in the combo list example:

```
var aImgSrc = 'Images/bullet_square_red.gif';
var aComboItems = new Array();
    aComboItems.push( {text:'<IMG align="middle" SRC="' + aImgSrc + '">
100,12$', value:100.12} );
    aComboItems.push( {text:'<IMG align="middle" SRC="' + aImgSrc + '">
1200$', value:1200} );
    aComboItems.push( {text:'<IMG align="middle" SRC="' + aImgSrc + '">
2000$', value:2000} );
    aComboItems.push( {text:'<IMG align="middle" SRC="' + aImgSrc + '">
5200$', value:5200} );
    aC1Edit.setComboItems(aComboItems);
```

Displaying Invalid Input Color

The following topic shows how to use the InvalidInputColor property for the C1MaskedInput control; however, the InvalidInputColor property is available for all **Input for ASP.NET AJAX** controls.

Using the Designer

To display a color when the user enters improper input, complete the following tasks:

1. Open the **C1MaskedInput Tasks** menu and click the ellipsis next to the InvalidInputColor property.
2. Select a color that will appear when the user enters improper input. For this example, select **Red**.

Using HTML Markup

To display a color when the user enters improper input, use the following markup in the .aspx page:

```
<cc1:C1MaskedInput ID="C1MaskedInput1" runat="server"
      Mask="(999) 000-0000"
      Text="412"
      InvalidInputColor="Red">
</cc1:C1MaskedInput>
```

Using Code

To display a color when the user enters improper input, double-click the Web page to create an event handler for the **Load** event. Enter the following code for the **Page_Load** event

- Visual Basic

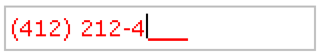
```
Me.C1MaskedInput1.Mask = "(999) 000-0000"
Me.C1MaskedInput1.Text = "412"
' Set invalid input color
Me.C1MaskedInput1.InvalidInputColor = Drawing.Color.Red
```

- C#

```
this.C1MaskedInput1.Mask = "(999) 000-0000";
this.C1MaskedInput1.Text = "412";
// Set invalid input color
this.C1MaskedInput1.InvalidInputColor = Drawing.Color.Red;
```

This topic illustrates the following:

Run the project and type a non-numeric key, which is invalid, and notice that the control displays red text:

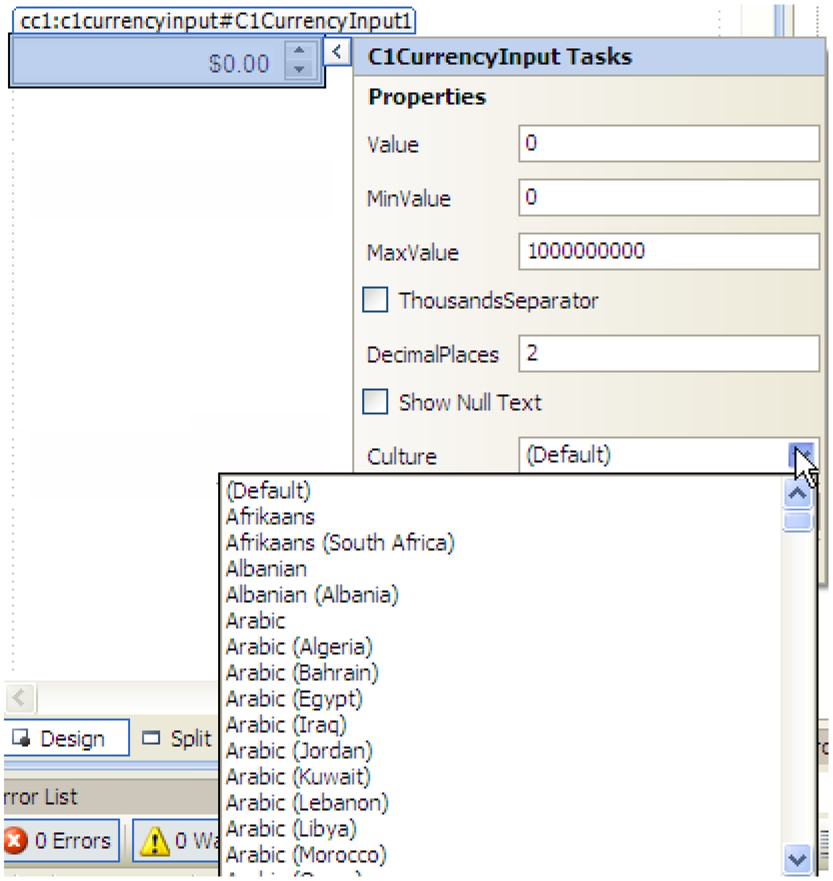

(412) 212-4___

Selecting the Culture

Note that the following topic shows how to use the Culture property for the **C1CurrencyInput** control; however, the Culture property is available for all **Input for ASP.NET AJAX** controls.

Using the Designer

You can choose a specific culture for any of the **Input for ASP.NET AJAX** controls. To set the Culture property for the control, simply open its **Tasks** menu and select a culture from its drop-down list.



Using HTML Markup

To set the Culture value, use the following markup in the .aspx page:

```
<cc1:C1CurrencyInput ID="C1CurrencyInput1" runat="server"
      Culture="English (United Kingdom)">
</cc1:C1CurrencyInput>
```

Using Code

To set the Culture for the **C1CurrencyInput** control, double-click the Web page to create an event handler for the **Load** event. Enter the following code for the **Page_Load** event:

- Visual Basic

```
Me.C1CurrencyInput1.Culture = New System.Globalization.CultureInfo("en-
GB")
```

- C#

```
this.C1CurrencyInput1.Culture = new System.Globalization.CultureInfo("en-
GB");
```

This topic illustrates the following:

The following **C1CurrencyInput** control shows the British Pound: