
ComponentOne

Input for ASP.NET Wijmo

Copyright © 2012 ComponentOne LLC. All rights reserved.

Corporate Headquarters

ComponentOne LLC

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

Internet: info@ComponentOne.com

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

Sales

E-mail: sales@componentone.com

Telephone: 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

Trademarks

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

Warranty

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

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

Copying and Distribution

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

This manual was produced using [ComponentOne Doc-To-Help™](#).

Table of Contents

ComponentOne Input for ASP.NET Wijmo Overview	1
Installing Studio for ASP.NET Wijmo	2
Studio for ASP.NET Wijmo Setup Files	2
System Requirements	2
Uninstalling Studio for ASP.NET Wijmo	3
Deploying your Application in a Medium Trust Environment	3
End-User License Agreement	6
Licensing FAQs	6
What is Licensing?	6
How does Licensing Work?	6
Common Scenarios	7
Troubleshooting	9
Technical Support	11
Redistributable Files	11
About This Documentation	12
Namespaces	12
Creating an ASP.NET Project	13
Adding the Input for ASP.NET Wijmo Components to a Project	14
Migrating from ASP.NET AJAX to ASP.NET Wijmo	15
Creating a ComponentOne Input for ASP.NET AJAX Project to Migrate	15
Creating the Project with ComponentOne Input for ASP.NET Wijmo	19
Key Features	23
Wijmo Top Tips	24
Input for ASP.NET Wijmo Quick Start	25
Step 1 of 5: Add Input for ASP.NET Wijmo Controls to Your Form	25
Step 2 of 5: Change the Appearance of Your Input for ASP.NET Wijmo Controls	26
Step 3 of 5: Format Your Input for ASP.NET Wijmo Controls	26
Step 4 of 5: Add a Culture Setting	28
Step 5 of 5: Run Your Quick Start Web Application	30
Design-Time Support	33

Input for ASP.NET Wijmo Smart Tags.....	33
C1InputMask Smart Tag	33
C1InputDate Smart Tag.....	35
C1InputNumeric Smart Tag.....	36
C1InputPercent Smart Tag	38
C1InputCurrency Smart Tag	39
Input for ASP.NET Wijmo Context Menus.....	41
Input for ASP.NET Wijmo Designers	41
C1InputMask Designer.....	41
C1InputDate Designer	42
Using C1InputMask.....	43
Defining C1InputMask.....	44
Using C1InputDate	46
Defining C1InputDate.....	47
Using C1InputNumeric	49
Defining C1InputNumeric	49
Using C1InputPercent	50
Defining C1InputPercent	51
Using C1InputCurrency	51
Defining C1InputCurrency	51
Input for ASP.NET Wijmo Appearance	52
Built-in Wijmo Themes	52
ThemeRoller Overview	53
C1Input CSS Selectors	53
Working with the Client-Side	54
Client-Side Events.....	54
Input for ASP.NET Wijmo Samples	57
Input for ASP.NET Wijmo Task-Based Help	58
C1InputMask Tasks.....	58
Changing the Prompt Character.....	58
Using Password Protection.....	59
Creating Day of the Week Chooser Mask.....	60
Creating an IP Address Mask.....	60
Creating a Phone Number Mask.....	62

Displaying the Date Mask without Prompt Characters.....	62
Hiding the Prompt Character on Leave.....	63
C1InputDateTasks.....	64
Setting the Date Format Pattern and Date.....	64
Displaying an Empty Date Value.....	66
C1InputNumeric Tasks.....	67
Indicating the Number of Decimal Places.....	67
Setting the Min/Max Value.....	68
Changing the Theme.....	68
Adding a Custom Theme.....	69
Selecting the Culture.....	71
Client-Side Event Tasks.....	72
Showing a Tooltip when Invalid Input is Entered.....	72
Using a Trigger to Show a Custom UI.....	72
Input for ASP.NET Wijmo Client-Side Reference.....	75
Using the Wijmo CDN.....	75

ComponentOne Input for ASP.NET Wijmo Overview

Your complete collection of data-entry and validation controls: choose from masked, date, numeric, and custom editing. Get built-in masks, custom format support, localization, and more with **ComponentOne Input™ for ASP.NET Wijmo**.

Getting Started

To get started, review the following topics:

- [Key Features](#) (page 23)
- [Input for ASP.NET Wijmo Quick Start](#) (page 25)

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

- **C1InputCurrency**

The C1InputCurrency control, derived from **C1InputNumeric**, 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 C1InputCurrency](#) (page 51).
- **C1InputDate**

The C1InputDate control, derived from **C1InputMask**, is specialized for editing the date and time. The **C1InputDate** control renders a date editor.

For details, see [Using C1InputDate](#) (page 46).
- **C1InputMask**

The C1InputMask 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, **C1InputMask** also serves as the base class for specialized controls such as **C1InputDate** and **C1InputNumeric**.

For details, see [Using C1InputMask](#) (page 43).
- **C1InputNumeric**

The C1InputNumeric control, derived from **C1InputMask**, 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 C1InputNumeric](#) (page 49).
- **C1InputPercent**

The C1InputPercent control, derived from **C1InputNumeric**, 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 C1InputPercent](#) (page 50).

For a list of the latest features added to **ComponentOne Studio for ASP.NET Wijmo**, visit [What's New in Studio for ASP.NET Wijmo](#).

Installing Studio for ASP.NET Wijmo

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

Studio for ASP.NET Wijmo Setup Files

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

Bin	Contains copies of all binaries (DLLs, Exes) in the ComponentOne Visual Studio ASP.NET Wijmo package.
Wijmo	Contains files (at least a readme.txt) related to the product.

The **ComponentOne Studio for ASP.NET Wijmo Help Setup** program installs integrated Microsoft Help Viewer help to the C:\Program Files\ComponentOne\Studio for ASP.NET Wijmo\HelpViewer folder.

Samples

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

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

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

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

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

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

System Requirements

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

Operating Systems:	Windows Server® 2003 Windows Server 2008 Windows XP SP2 Windows Vista™ Windows 7
Web Server:	Microsoft Internet Information Services (IIS) 6.0 or later
Environments:	.NET Framework 3.0 or later Visual Studio 2008 or later Internet Explorer 6.0 or later

Firefox® 2.0 or later

Safari® 2.0 or later

Uninstalling Studio for ASP.NET Wijmo

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

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

To uninstall **Studio for ASP.NET Wijmo** integrated help:

1. Open the Control Panel and select **Add or Remove Programs** (Programs and Features in Windows 7/Vista).
4. Select **ComponentOne Studio for ASP.NET Wijmo Help** and click the **Remove** button.
5. Click **Yes** to remove the integrated help.

Deploying your Application in a Medium Trust Environment

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

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

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

Modifying or Editing the Config File

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

Edit the Config File

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

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

Create a Custom Policy Based on Medium Trust

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

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

Give the new a name that indicates that it is your variation of medium trust; for example, AllowReflection_Web_MediumTrust.config.

3. Add the permissions that you want to grant. For examples, see [Adding Permissions](#) (page 4).
4. Enable the custom policy file on your application by modifying the following lines in your web.config file under the <system.web> node:

```
<system.web>
  <trust level="CustomMedium" originUrl="" />

  <securityPolicy>
    <trustLevel name="CustomMedium"
policyFile="AllowReflection_Web_MediumTrust.config" />
  </securityPolicy>
  ...
</system.web>
```

Note: Your host may not allow trust level overrides. Please check with your host to see if you have these rights.

Allowing Deserialization

To allow the deserialization of the license added to App_Licenses.dll by the Microsoft IDE, you should add the SerializationFormatter flag to security permission to the Web.config file. Complete the steps in the [Modifying or Editing the Config File](#) (page 3) topic to create or modify a policy file before completing the following.

Add the `SerializationFormatter` flag to the `<IPermission class="SecurityPermission">` tag so that it appears similar to the following:

```
<NamedPermissionSets>
  <PermissionSet
    class="NamedPermissionSet"
    version="1"
    Name="ASP.Net">
    <IPermission
      class="SecurityPermission"
      version="1"
      Flags="Assertion, Execution, ControlThread, ControlPrincipal,
RemotingConfiguration, SerializationFormatter" />
    ...
  </PermissionSet>
</NamedPermissionSets>
```

Adding Permissions

You can add permission, including ReflectionPermission, OleDbPermission, and FileIOPermission, to the web.config file. Note that ComponentOne controls will not work in an environment where reflection is not allowed. Complete the steps in the [Modifying or Editing the Config File](#) (page 3) topic to create or modify a policy file before completing the following.

ReflectionPermission

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

To add reflection permission, complete the following:

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

```

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

```

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

```

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

```

4. Save and close the `web_mediumtrust.config` file.

OleDbPermission

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

To add `OleDbPermission`, complete the following steps:

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

```

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

```

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

```

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

```

4. Save and close the `web_mediumtrust.config` file.

FileIOPermission

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

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

1. Open the `web_mediumtrust.config` file or a file created based on the `web_mediumtrust.config` file.

2. Add the following `<SecurityClass>` tag after the `<SecurityClasses>` tag so that it appears similar to the following:

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

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

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

4. Save and close the `web_mediumtrust.config` file.

End-User License Agreement

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

Licensing FAQs

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

What is Licensing?

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

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

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

How does Licensing Work?

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

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

When a user decides to purchase a product, he receives an installation program and a Serial Number. During the installation process, the user is prompted for the serial number that is saved on the system. (Users can also enter

the serial number by clicking the **License** button on the **About Box** of any ComponentOne product, if available, or by rerunning the installation and entering the serial number in the licensing dialog box.)

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

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

These files are automatically added to the project.

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

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

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

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

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

Common Scenarios

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

Creating components at design time

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

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

Creating components at run time

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

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

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

Adding an instance of the component to a form, then removing that component, is just a simple way of adding a line with the component strong name to the licenses.licx file. If desired, you can do this manually using notepad or

Visual Studio itself by opening the file and adding the text. When Visual Studio recreates the application resources, the component will be queried and its run-time license added to the appropriate assembly resource.

Inheriting from licensed components

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

- Add a LicenseProvider attribute to the component.

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

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

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

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

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

Using licensed components in console applications

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

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

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

Using licensed components in Visual C++ applications

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

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

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

```
c1lc /target:MyApp.exe /complist:licenses.licx  
/i:C1.Win.C1FlexGrid.dll
```

5. Link the licenses into the project. To do this, go back to Visual Studio, right-click the project, select properties, and go to the Linker/Command Line option. Enter the following:
`/ASSEMBLYRESOURCE:Debug\MyApp.exe.licenses`
6. Rebuild the executable to include the licensing information in the application.

Using licensed components with automated testing products

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

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

For example:

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

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

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

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

Troubleshooting

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

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

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

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

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

If that fails follow these steps:

1. Open the affected project.
2. Select an instance of the updated component.
3. In the Visual Studio Properties window, change any property. It does not matter which property you change; you can change it back to the previous value.
4. Rebuild the project using the **Rebuild All** option (not just **Rebuild**) and run the solution.

Alternative 1: Follow these steps:

1. Open a new Visual Studio.NET project.

2. Add the updated component to the form.
3. Compile and run the new project.
4. Open the licenses.licx file in the new project.
5. Copy the line that starts with the namespace of the updated component (for example, C1.Win.C1Report) and ends with a public key token.
6. Open the existing, incorrectly licensed project.
7. Open the licenses.licx file in the new project.
8. Paste the line from step 5 into this file (replace the old licensing information with the new).
9. Rebuild the project using the **Rebuild All** option (not just **Rebuild**) and run the solution.

Alternative 2: Follow these steps:

1. Open the affected project.
2. Delete the licenses.licx file from the project.
3. Add a new instance of the updated component to the form.
4. Rebuild and run the solution. The nag screen should not appear.
5. Remove the newly added component from the form.

Try each of these options multiple times, if necessary. If that still does not help, are you creating any of the controls in code rather than design-time? If so, you must add an entry for the control in the licenses.licx file (see <http://helpcentral.componentone.com/PrintableView.aspx?ID=1886> for more information). Also if this is a website, as opposed to an ASP.NET web application, please try right-clicking the licenses.licx file and selecting "Build Runtime Licenses" from the context menu.

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

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

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

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

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

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

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

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

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

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

Option 2 – Continue to use the components you have.

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

Technical Support

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

Some methods for obtaining technical support include:

- **[Online Resources](#)**
ComponentOne provides customers with a comprehensive set of technical resources in the form of FAQs, samples and videos, Version Release History, searchable Knowledge base, searchable Online Help and more. We recommend this as the first place to look for answers to your technical questions.
- **Online Support via our Incident Submission Form**
This online support service provides you with direct access to our Technical Support staff via an [online incident submission form](#). When you submit an incident, you'll immediately receive a response via e-mail confirming that you've successfully created an incident. This email will provide you with an Issue Reference ID and will provide you with a set of possible answers to your question from our Knowledgebase. You will receive a response from one of the ComponentOne staff members via e-mail in 2 business days or less.
- **Product Forums**
ComponentOne's [product forums](#) are available for users to share information, tips, and techniques regarding ComponentOne products. ComponentOne developers will be available on the forums to share insider tips and technique and answer users' questions. Please note that a ComponentOne User Account is required to participate in the ComponentOne Product Forums.
- **Installation Issues**
Registered users can obtain help with problems installing ComponentOne products. Contact technical support by using the [online incident submission form](#) or by phone (412.681.4738). Please note that this does not include issues related to distributing a product to end-users in an application.
- **Documentation**
Microsoft integrated ComponentOne documentation can be installed with each of our products, and documentation is also available online. If you have suggestions on how we can improve our documentation, please email the [Documentation team](#). Please note that e-mail sent to the [Documentation team](#) is for documentation feedback only. [Technical Support](#) and [Sales](#) issues should be sent directly to their respective departments.

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

Redistributable Files

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

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

- C1.Web.Wijmo.Extenders.3.dll
- C1.Web.Wijmo.Extenders.4.dll
- C1.C1Report.2.dll
- C1.C1Report.4.dll

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

About This Documentation

Acknowledgements

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

Firefox is a registered trademark of the Mozilla Foundation.

Safari is a registered trademark of Apple Inc.

ComponentOne

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

Corporate Headquarters

ComponentOne LLC

201 South Highland Avenue

3rd Floor

Pittsburgh, PA 15206 • USA

412.681.4343

412.681.4384 (Fax)

<http://www.componentone.com>

ComponentOne Doc-To-Help

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

Namespaces

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

The general namespace for ComponentOne Web products is **C1.Web**. The following code fragment shows how to declare a **C1InputMask** using the fully qualified name for this class:

- Visual Basic

```
Dim MaskedInput As C1.Web.Wijmo.Controls.C1Input.C1InputMask
```

- C#

```
C1.Web.Wijmo.Controls.C1Input.C1InputMask 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 C1InputMask = C1.Web.UI.Controls.C1Input.C1InputMask
Imports MyInputMask = MyProject.Objects.C1Input.C1InputMask

Dim wm1 As C1InputMask
Dim wm2 As MyInputMask
```

- C#

```
using C1InputMask = C1.Web.UI.Controls.C1Input.C1InputMask;
using MyInputMask = MyProject.Objects.C1Input.C1InputMask;

C1InputMask wm1;
MyInputMask wm2;
```

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

Creating an ASP.NET Project

ComponentOne Studio for ASP.NET Wijmo requires Visual Studio 2008 or later and .NET Framework 3.0 or later for your Web applications. **Studio for ASP.NET Wijmo** does not require AJAX extensions; however, you can install them if you want to use AJAX in your project. See the following table for more details on installing AJAX extensions.

Visual Studio 2010	You can build Ajax-enabled ASP.NET projects by downloading the AJAX Control Toolkit from CodePlex and dragging-and-dropping the controls from the Visual Studio Toolbox onto an ASP.NET Web Forms page.
Visual Studio 2008, .NET Framework 3.5	You can easily create an AJAX-enabled ASP.NET project without installing separate add-ins because the framework has a built-in AJAX library and controls.
Visual Studio 2008, .NET Framework 3.0	You must install the ASP.NET AJAX Extensions 1.0, which can be found at http://www.asp.net/ajax/downloads/archive/ . Then you can create an AJAX 1.0-Enabled ASP.NET 2.0 Web site or application.

The following topics explain how to create projects in Visual Studio 2010 and 2008.

- **Creating a Web Site/Application Project in Visual Studio 2010** 

To create a new Web site/application project in Visual Studio 2010, complete the following steps.

1. If you are creating an AJAX project, download the AJAX Control Toolkit from [CodePlex](#) and extract the files.
2. From the **File** menu, select **New | Web Site/Project**. The New Web Site/New Project dialog box opens.
3. Select a .NET Framework in the upper right corner. Note that if you choose .NET Framework 3.0, you must install the [extensions](#) first.

4. Under **Project Types**, choose either **Visual Basic** or **Visual C#** and then select **Web**. Note that one of these options may be located under **Other Languages**.
5. Select a language, and in the list of templates, select **ASP.NET Web Site/Application**.
6. Specify a location and then click **OK**.

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

A new Web project is created at the root of the Web server you specified.

7. If you are creating an AJAX project, right-click the Toolbox (create a new tab if you like), select **Choose Items** and browse to find the **AjaxControlToolkit.dll**. You can begin dragging toolkit controls to your page. Note that if you do not see the toolkit controls in the Toolbox, make sure you selected the .NET Framework that corresponds with the version of the toolkit you downloaded.

- **Creating a Web Site/Application Project in Visual Studio 2008** 

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

1. From the **File** menu, select **New | Web Site/Project**. The New Web Site/New Project dialog box opens.
2. Select .NET Framework 3.5 or 3.0 in the upper right corner. Note that if you choose .NET Framework 3.0, you must install the [extensions](#) first.
3. Select a language, and in the list of templates, select **ASP.NET Web Site/Application** or **AJAX 1.0-Enabled ASP.NET 2.0 Web Site/Application**.
4. Specify a location and then click **OK**.

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

A new Web project is created at the root of the Web server you specified.

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

When you open Visual Studio, you will notice a **ComponentOne Studio for ASP.NET Wijmo Projects** tab containing the ComponentOne controls that have automatically been added to the Toolbox.

Note that you can manually add ComponentOne controls to the Toolbox at a later time.

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

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

- C1InputDate
- C1InputMask
- C1InputCurrency
- C1InputNumeric
- C1InputPercent

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

1. Open the Visual Studio IDE (Microsoft Development Environment). Make sure the Toolbox is visible (select **Toolbox** in the **View** menu if necessary) and right-click it to open the context menu.

2. To make the Studio for ASP.NET Wijmo components appear on their own tab in the Toolbox, select **Add Tab** from the context menu and type in the tab name, Studio for ASP.NET Wijmo, for example.
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.Wijmo.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 Wijmo Controls to the Form

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

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

Adding a Reference to the Assembly

To add a reference to the C1.Web.Wijmo.Controls.3 or C1.Web.Wijmo.Controls.4 assembly:

1. Select the **Add Reference** option from the **Website** menu of your Web Site project or from the **Project** menu of your Web Application project.
2. Select the most recent version of the **ComponentOne Studio for ASP.NET Wijmo** assembly from the list on the **NET** tab or browse to find the C1.Web.Wijmo.Controls.3.dll or C1.Web.Wijmo.Controls.4.dll file and click **OK**.
3. Select the **Form1.vb** tab or go to **View | Code** to open the Code Editor. At the top of the file, add the following **Imports** directive (**using** in C#):

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

Note: This makes the objects defined in the **C1.Web.Wijmo.Controls.3(4)** assembly visible to the project. See [Namespaces](#) (page 12) for more information.

Migrating from ASP.NET AJAX to ASP.NET Wijmo

Completely redesigned to take full advantages of the latest Web technologies such as HTML5, JQuery, and CSS3, **ComponentOne Studio for ASP.NET Wijmo** has improved upon and effectively replaced the ComponentOne ASP.NET AJAX control suite.

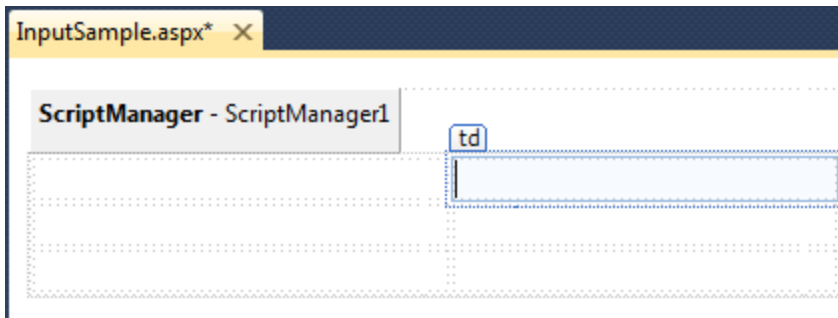
Converting your **ComponentOne Input for ASP.NET AJAX (C1Input)** projects from AJAX to Wijmo is simple. The following topics will show you the differences between the AJAX and Wijmo versions of the **C1Input** controls so you can quickly migrate and take advantage of Wijmo.

First we'll create a small sample application using **ComponentOne Input for ASP.NET AJAX** and then show how the sample can be created in **ComponentOne Input for ASP.NET Wijmo**.

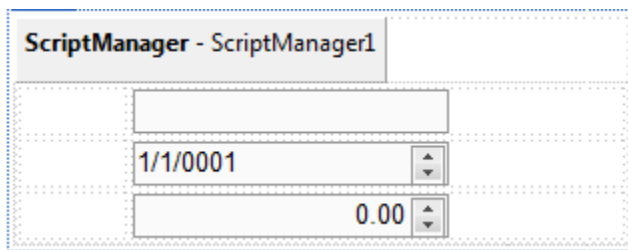
Creating a ComponentOne Input for ASP.NET AJAX Project to Migrate

The first thing we need to do is create a small ASP.NET AJAX sample that we can migrate to Wijmo.

1. Create a new Empty ASP.NET Web App in Visual Studio.
2. Select **View | Designer** and drag a **ScriptManager** control from the Toolbox onto the page.
3. Click **Table | Insert Table** in the Visual Studio menu and make a table with 3 rows and 2 columns, like the following example:

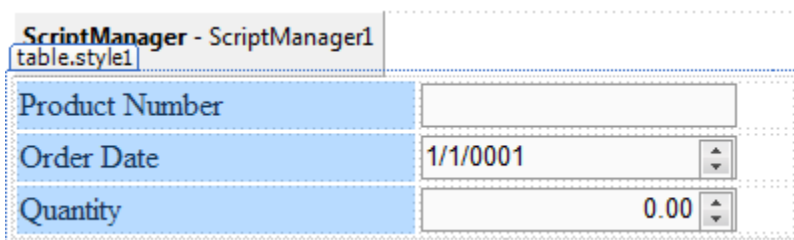


4. Before we can add the controls to the page, we need to add some assembly references to the project. Select **Project | Add Reference**, and browse to find and select the following assemblies:
 - C1.Web.UI.4.dll
 - C1.Web.UI.Controls.4.dll
 - C1.Web.UI.Design.4.dll
5. Once you have those references added, it's time to start adding controls. Start by adding **C1MaskedInput** control to the table in the top right position, **C1DateInput** in the middle right position and **C1NumericInput** in the bottom right position. Once all of the controls have been added, your page should look similar to this:



For the moment, don't worry about the spacing of the controls in the table as you will be able to adjust this shortly.

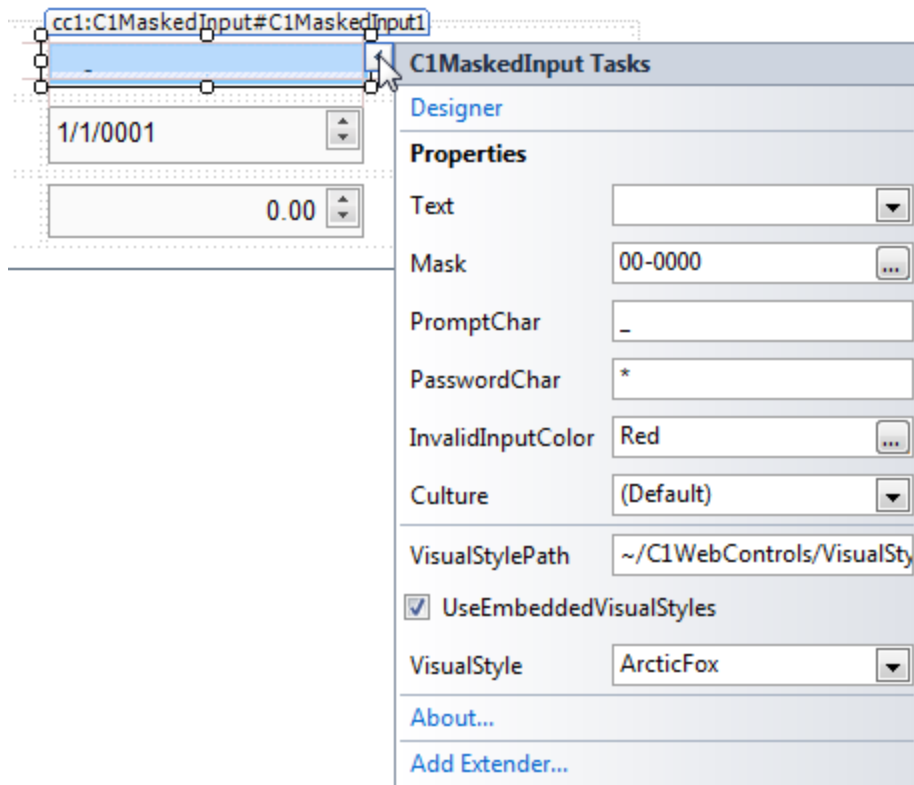
6. The next step is to add some text to the left side of the table to let the user know what the controls are displaying. Start by adding the text **Product Number** in the top left position on the table. Then add the text **Order Date** in the middle left position of the table. Finally, add the text **Quantity** in the bottom left cell of the table. The end result should look similar to this:



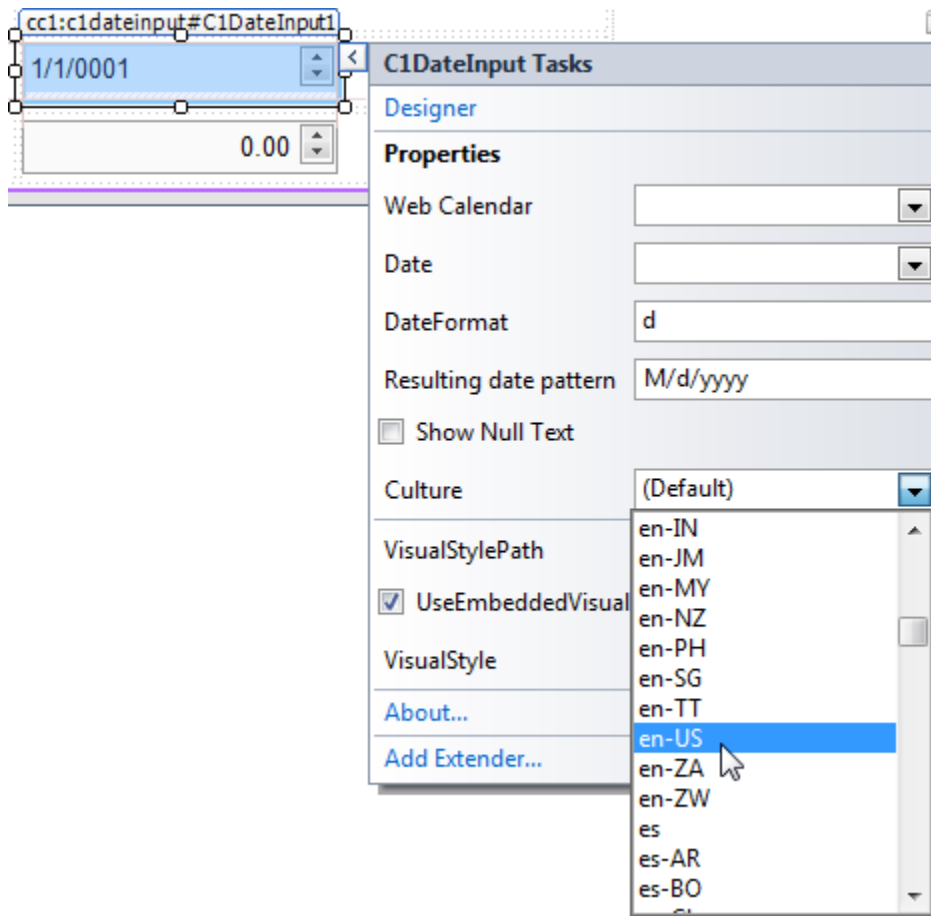
Setting Properties for AJAX Controls

Next we're going to set some properties to get the controls to behave like they are supposed to. First let's start with the **C1MaskedInput**. Each of the controls has a **Task** menu that can be accessed by clicking the smart tag, or the small arrow located at the top right of each control.

1. Select the **C1MaskedInput** control and click the smart tag to open the **C1MaskedInput Tasks** menu.
2. Set the **Mask** property to **00-0000**, the **PasswordChar** property to ***** and the **InvalidInputColor** property to **Red**.



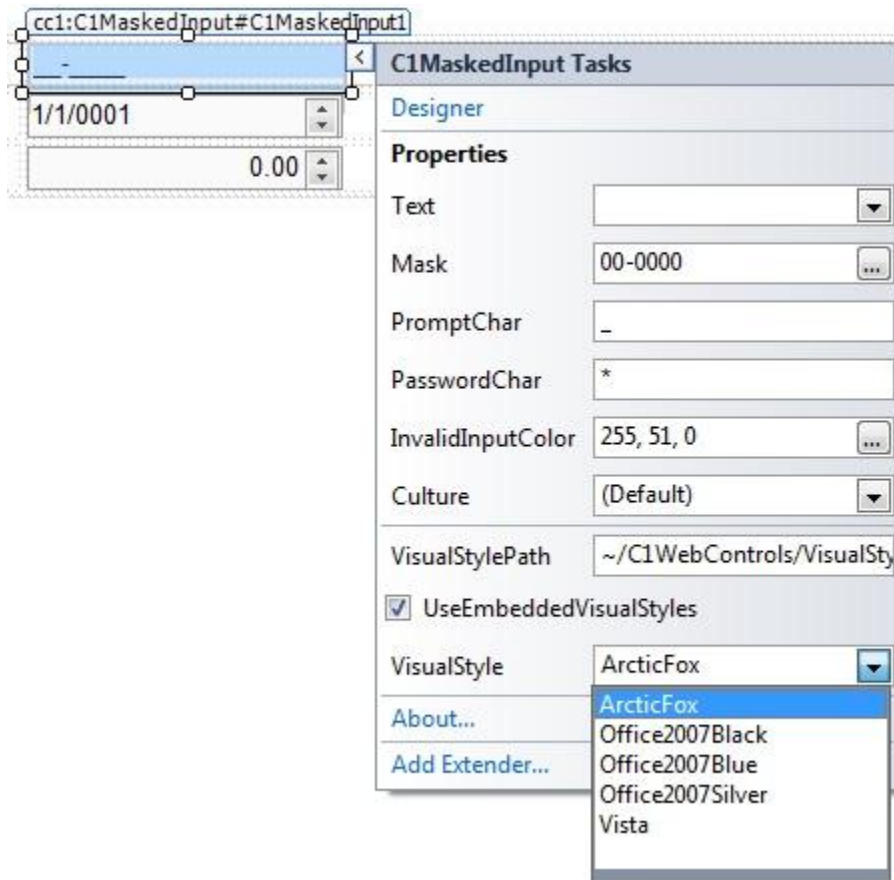
3. Next select the **C1DateInput** control and click the smart tag to open the **C1DateInput Tasks** menu. Change the properties according to your preferences. Keep in mind that you can change the culture and format to suit any cultural needs. For instance, if you wanted your **C1DateInput** control to suit French end-users, then you would change the culture property to **fr-FR**. But for this sample, set the culture to **en-US** which is the culture setting for the United States.



4. Now select the **C1NumericInput** control. In the **C1NumericInput Tasks** menu, set the **minValue** to 0 and the **maxValue** to 10. Set the **Culture** property to **en-US** for the United States culture setting.

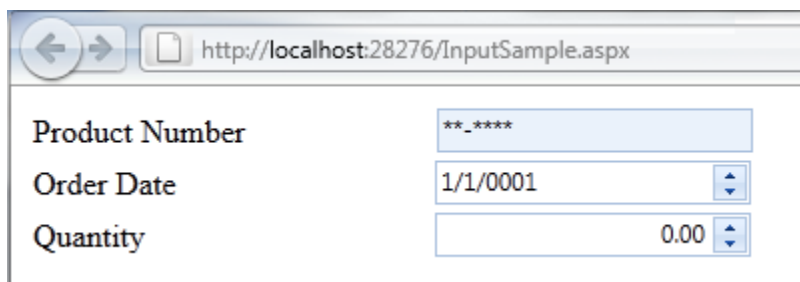
Visual Styles

Now that we have the basic foundation for our application, let's give it a more appealing look and feel. From the **Tasks** menus for each of the controls, set the **VisualStyle** property to one of the style options: *ArcticFox*, *Office2007Black*, *Office2007Blue*, *Office2007Silver*, or *Vista*.



Running the AJAX Application

Save the application and press F5 to run it. It will look similar to the following:



Creating the Project with ComponentOne Input for ASP.NET Wijmo

Now that we have a finished base application, we can get started migrating to Wijmo! Oh the anticipation. One important thing to remember starting out is that Wijmo controls do not need a **ScriptManager** to function properly.

One of the main differences between the AJAX and Wijmo versions of the **C1Input** controls is the naming of the controls. The following table lists the control names in both platforms. Keep the control names in mind when you convert your projects to Wijmo.

AJAX	Wijmo
C1CurrencyInput	C1InputCurrency

C1DateInput	C1InputDate
C1MaskedInput	C1InputMask
C1NumericInput	C1InputNumeric
C1PercentInput	C1InputPercent

We are going to build the same table with the same layout as we did in [Creating a ComponentOne Input for ASP.NET AJAX Project to Migrate](#) (page 15) topic.

1. Create a new Empty ASP.NET Web App in Visual Studio.
2. Select **View | Designer** to switch to Design view.
3. Click **Table | Insert Table** in the Visual Studio menu and make a table with 3 rows and 2 columns.
4. Before we can add the controls to the page, we need to add some assembly references to the project. Select **Project | Add Reference**, and browse to find and select the following assemblies:
 - C1.Web.Wijmo.Controls.4.dll
 - C1.Web.Wijmo.Design.4.dll
5. Once you have those references added, it's time to start adding controls. Place the C1InputMask control in the top right cell, the C1InputDate control in the middle right cell, and the C1InputNumeric control in the bottom right cell.
6. Add the text **Product Number** in the top left cell, **Order Date** in the middle left cell, and **Quantity** in the bottom left cell. The end result should look similar to this:

Product Number	<input type="text"/>
Order Date	9/15/2011
Quantity	0.00

Setting Properties for Wijmo Controls

The Wijmo controls have a few different options than their predecessors. For instance, select the C1InputMask control and click the smart tag to open the **C1InputMask Tasks** menu. You will find theming options instead of visual styles. The themes are global themes, meaning that if you change the theme of one control on your page, all of the controls will follow suit.

1. Set the **Mask** property to **00-0000**, the **PasswordChar** property to ***** and the **Theme** property to **rocket**.
2. Select the C1InputDate control and click the smart tag to open the **C1InputDate Tasks** menu. Notice that the theme has automatically changed to **rocket** since the theme was set in the C1InputMask control.
3. Set the culture to **en-US**, which is the culture setting for the United States.
4. Now select the C1InputNumeric control. In the **C1InputNumeric Tasks** menu, set the **minValue** to **0** and the **maxValue** to **10**. Set the **Culture** property to **en-US** for the United States culture setting.

Running the Wijmo Application

Press **F5** to run the application and check out what we've done.

Product Number	<input type="text" value="**_****"/>
Order Date	<input type="text" value="9/15/2011"/>
Quantity	<input type="text" value="0.00"/>

The theming options of the new Wijmo controls go above and beyond what is offered in the AJAX controls and allow the user to develop rich, aesthetically appealing applications.

ASPX Markup Comparison

Here is the aspx markup for both the AJAX and Wijmo projects to show you the slight differences between the two platforms.

AJAX:

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.master"
AutoEventWireup="true"
CodeBehind="Default.aspx.cs" Inherits="InputMigration. Default" %>

<%@ Register assembly="Cl.Web.UI.Controls.4" namespace="Cl.Web.UI.Controls.ClInput"
tagprefix="cc1" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
  <style type="text/css">
    .style1
    {
      width: 54%;
      height: 115px;
    }
    .style2
    {
      height: 35px;
    }
    .style3
    {
      height: 35px;
      width: 208px;
    }
    .style4
    {
      width: 208px;
    }
    .style5
    {
      height: 36px;
      width: 208px;
    }
    .style6
    {
      height: 36px;
    }
  </style>
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
  <h2>
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
  </h2>
  <p>
    &nbsp;  </p>
  <table class="style1">
    <tr>
      <td class="style3">
```

```

        Product Number</td>
        <td class="style2">
            <cc1:C1MaskedInput ID="C1MaskedInput1" runat="server" Width="155px"
                InvalidInputColor="Red" Mask="00-0000" PasswordChar="*" />
        </td>
    </tr>
    <tr>
        <td class="style5">
            Order Date</td>
        <td class="style6">
            <cc1:C1DateInput ID="C1DateInput1" runat="server" Culture="en-US"
                style="top: 0px; left: 0px; height: 1.8em" Width="155px" />
        </td>
    </tr>
    <tr>
        <td class="style4">
            Quantity</td>
        <td>
            <cc1:C1NumericInput ID="C1NumericInput1" runat="server" Culture="en-US"
                MaxValue="10" style="top: 0px; left: 0px; height: 1.6em"
Width="155px" />
        </td>
    </tr>
</table>
</asp:Content>

```

Wijmo:

```

<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.master"
AutoEventWireup="true"
CodeBehind="Default.aspx.cs" Inherits="InputMigrationWijmo. Default" %>

<%@ Register assembly="C1.Web.Wijmo.Controls.4" namespace="C1.Web.Wijmo.Controls.C1Input"
tagprefix="wijmo" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
    <style type="text/css">
        .style1
        {
            width: 37%;
            height: 96px;
        }
        .style2
        {
            width: 145px;
        }
    </style>
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <h2>
        &nbsp;  </h2>
    <table class="style1">
        <tr>
            <td class="style2">
                Product Number</td>
            <td>
                <wijmo:C1InputMask ID="C1InputMask1" runat="server" Mask="00-0000"
                    PasswordChar="*">
                </wijmo:C1InputMask>
            </td>
        </tr>
        <tr>
            <td class="style2">
                Order Date</td>
            <td>
                <wijmo:C1InputDate ID="C1InputDate1" runat="server">
                </wijmo:C1InputDate>
            </td>
        </tr>
    </table>

```

```

        <td class="style2">
            Quantity</td>
        <td>
            <wijmo:C1InputNumeric ID="C1InputNumeric1" runat="server" MaxValue="10"
                MinValue="0">
            </wijmo:C1InputNumeric>
        </td>
    </tr>
</table>
</asp:Content>

```

Key Features

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

- **Rich Client-side Object Model**

Create a richer user experience on the client with mouse-action and control-focus events. Change the control format, color, mask, minimum and maximum values and more with multiple client-side methods.
- **Over 23 Built-in Masks**

Select from over 23 built-in masks or customize your own. The C1InputMask control includes 14 built-in standard masks, including time and date formats, day of week chooser, and numeric range. The C1InputDate control includes 9 built-in standard masks, including short and long date and time formats. Both include custom format support.
- **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.
- **Rich Formatting Model**

Format input boxes in almost any way imaginable. A rich formatting model enables developers to customize the appearance of a control's text, border, cell spacing, color scheme, and so on.
- **Design-time Support**

Use the **SmartTag** to quickly access the five control-specific designers. Set masks, format, value, and culture properties, and visualize your edits with the WYSIWYG window. See [Design-Time Support](#) (page 33) for more information.
- **Cultural Support**

Define the cultural setting used by any input control – this applies to string comparison, numeric and date time formats, and special characters. See [Selecting the Culture](#) (page 71) for details.
- **Drop-down and Spin Buttons**

The specialized Input controls for date/time and numeric editing, C1InputDate, C1InputCurrency, and C1InputNumeric controls support drop-down and spin (up/down) buttons.
- **Date Picker**

A calendar may be used as a date picker in the C1InputDate control. To enable the trigger button to open the default calendar, simply set the ShowTrigger property to **True**.
- **Numeric Editing**

Set value input limits, specify number of decimal places, and indicate use of the thousands separator for the C1InputNumeric, C1InputCurrency, and C1InputPercent controls.
- **Password Protection**

Using the C1InputMask control, protect input text by displaying password characters. At design time, simply select a character (*, for example) to substitute for the actual input characters. See [Using Password Protection](#) (page 59) for an example.

- **Keyboard Support**

Quickly edit input values using keyboard support. Move the cursor one position to the left or right or to the beginning or end, increment or decrement the range value, copy and paste, and more!

- **Theming**

With just a click of the **SmartTag**, change the input control's look by selecting one of the 6 premium themes (*Arctic*, *Midnight*, *Aristo*, *Rocket*, *Cobalt*, and *Sterling*). Optionally, use ThemeRoller from jQuery UI to create a customized theme! See [Input for ASP.NET Wijmo Appearance](#) (page 52) for additional information.

- **CSS Support**

Use a cascading style sheet (CSS) style to define custom skins. You have complete control over the input box's background, colors, borders, styles, padding, cell spacing, and more. CSS support allows you to match the input controls to your organization's standards.

Wijmo Top Tips

The following tips may help you troubleshoot when working with Studio for ASP.NET Wijmo.

Tip 1: Prevent poor page rendering in quirks mode by editing the meta tag to fix rendering.

If a user's browser is rendering a page in quirks mode, widgets and controls may not appear correctly on the page. This is indicated by a broken page icon in the address bar. In **Compatibility View**, the browser uses an older rendering engine.



Users can set this view that causes the issue. To prevent rendering in quirks mode, you can force the page to render with the latest browser. Add the following meta tag to the header of the page:

```
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
```

Input for ASP.NET Wijmo Quick Start

This section will lead you through the creation of a Web form that uses the **ComponentOne Input for ASP.NET Wijmo** 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 C1InputMask, C1InputDate, and C1InputCurrency controls. The C1InputNumeric and C1InputPercent controls will not appear in this Quick Start since they share similar properties to the C1InputCurrency control.

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

To begin, [create an ASP.NET Web Site](#) (page 13) and add the [Input for ASP.NET Wijmo](#) (page 14) controls to your Toolbox.

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 Wijmo** 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:
 - C1InputMask
 - C1InputDate
 - C1InputCurrency

The table should look similar to the following image:

	6/30/2011
	\$0.00

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:

Product Number:	
Order Date:	6/30/2011
Unit Price	\$0.00

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 Wijmo** 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 Wijmo Controls

This topic shows how to change the appearance of your **Input for ASP.NET Wijmo** 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, **C1InputMask**, and click the smart tag (☒). The **C1InputMask Tasks** menu appears.
3. From the **Tasks** menu, click the drop-down arrow next to **Theme** and select **rocket**. The other **C1Input** controls will also be updated with the **rocket** theme. The following image shows the appearance of the updated controls:



Product Number:	<input type="text"/>
Order Date:	6/30/2011
Unit Price	\$0.00

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

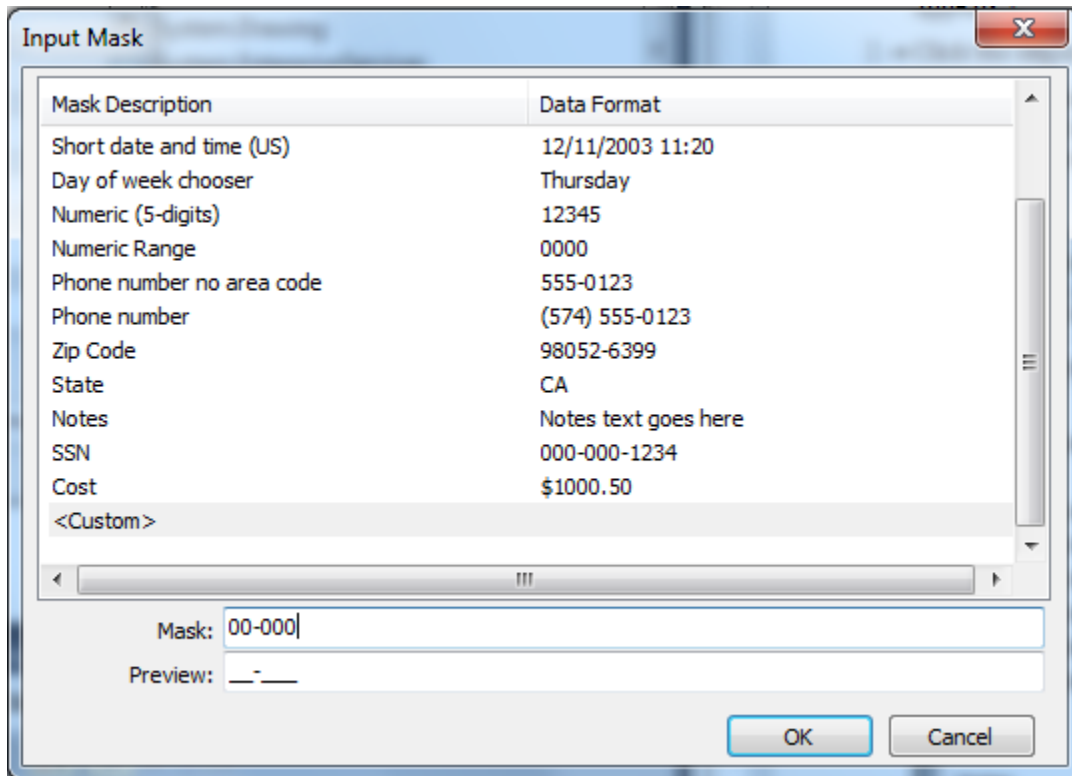
You have successfully changed the appearance of the **Input for ASP.NET Wijmo** controls. The next topic shows how to format the input boxes.

Step 3 of 5: Format Your Input for ASP.NET Wijmo 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 Wijmo** controls on your Web form.

To format the **C1InputMask** control:

1. Select the **C1InputMask** control and click the smart tag (☒). The **C1InputMask 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.



4. Click **OK** to close the **Input Mask** dialog box.
5. Select **View | Properties Window** and notice **ui-state-error** next to the **InvalidClass** property. If a user enters invalid input, such as alphanumeric characters, the input color appears red identifying the invalid entry, as specified in the **ui-state-error** in the CSS.

To format the **C1InputDate** control:

1. Select the **C1InputDate** control and click the smart tag (🔗). The **C1InputDate Tasks** menu appears.
2. Enter a date format in the **DateFormat** text box. In this example, we'll use **D**. 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 **C1InputDate Tasks** menu, click the drop-down arrow next to **Date**, and select a date from the drop-down calendar.
4. Locate the ShowSpinner property in the Visual Studio Properties window, click the drop-down arrow and select **True**.

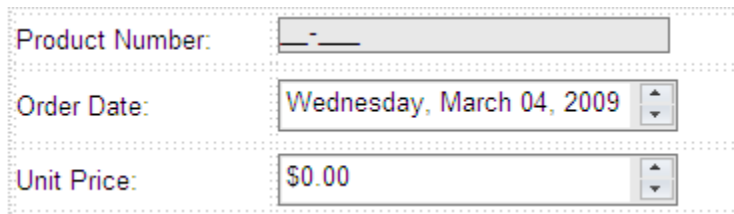
Note: You may have to resize the controls using the **Height** and **Width** properties in the Visual Studio Properties window.

To format the C1InputCurrency control:

1. Select the **C1InputCurrency** control and locate the ShowSpinner property in the Visual Studio Properties window.
2. Click the drop-down arrow and select **True**.

You have successfully changed the format of the **Input for ASP.NET Wijmo** 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**):



The next topic shows how to add buttons to change the culture information for the **C1InputDate** and **C1InputCurrency** 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 C1InputDate and C1InputCurrency 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 = 140px

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
    C1InputDate1.Culture = New System.Globalization.CultureInfo("fr-FR")
    C1InputCurrency1.Culture = New System.Globalization.CultureInfo("fr-FR")
End Sub
```

- C#

```
protected void FrenchBtn_Click(object sender, System.EventArgs e)
{
    C1InputDate1.Culture = new System.Globalization.CultureInfo("fr-FR");
    C1InputCurrency1.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
    C1InputDate1.Culture = New System.Globalization.CultureInfo("en-US")
    C1InputCurrency1.Culture = New System.Globalization.CultureInfo("en-US")
End Sub
```

- C#

```
protected void USEnglishBtn_Click(object sender, System.EventArgs e)
{
    C1InputDate1.Culture = new System.Globalization.CultureInfo("en-US");
    C1InputCurrency1.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:	<input type="text" value="-"/>
Order Date:	<input type="text" value="Wednesday, March 04, 2009"/>
Unit Price:	<input type="text" value="\$0.00"/>
<input type="button" value="French Culture"/> <input type="button" value="U.S. English Culture"/>	

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:	<input type="text" value="-"/>
Order Date:	<input type="text" value="Tuesday, July 05, 2011"/>
Unit Price:	<input type="text" value="\$0.00"/>
<input type="button" value="French Culture"/> <input type="button" value="U.S. English Culture"/>	

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 does not allow it.
- To change the Order Date (C1InputDate 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 (C1InputCurrency 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 C1InputDate and C1InputCurrency controls, click the **French Culture** button.
- To change the culture back to U.S. English for the C1InputDate and C1InputCurrency controls, click the **U.S. English Culture** button.

Congratulations!

You have successfully created a basic Web form with three different **Input for ASP.NET Wijmo** 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 Wijmo 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 Wijmo**.

Invoking the Tasks Menus

In Visual Studio, each control in **Input for ASP.NET Wijmo** 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 Wijmo**, see [Input for ASP.NET Wijmo Smart Tags](#) (page 33).

Invoking the Context Menus

You can easily configure any of the **Input for ASP.NET Wijmo** components at design time by using its associated context menu. For more information on **Input for ASP.NET Wijmo** context menus, see the [Input for ASP.NET Wijmo Context Menus](#) (page 41).

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

You can access the properties for any of **Input for ASP.NET Wijmo'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 Wijmo Smart Tags

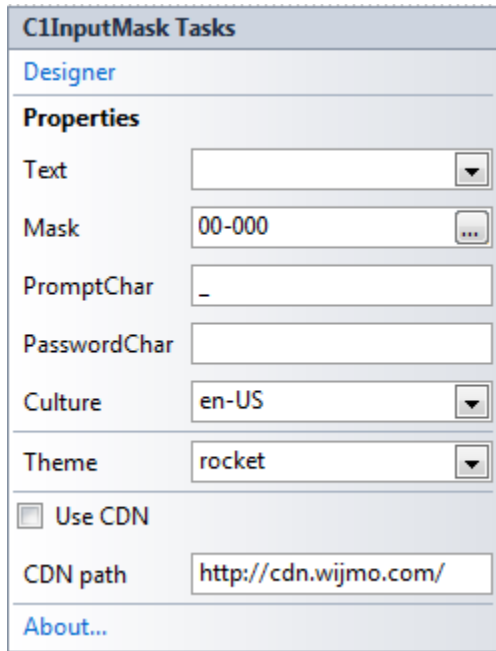
In Visual Studio, each control in **Input for ASP.NET Wijmo** includes a smart tag (☐). A smart tag represents a short-cut **Tasks** menu that provides the most commonly used properties in each control.

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

C1InputMask Smart Tag

The C1InputMask control provides quick and easy access to the most common C1InputMask properties through its smart tag.

To access the **C1InputMask Tasks** menu, click the smart tag (☐) in the upper-right corner of the C1InputMask control. This will open the **C1InputMask Tasks** menu.



The **C1InputMask Tasks** menu operates as follows:

Designer

Clicking **Designer** opens the **C1InputMask Designer**. For more information on the designer, see [C1InputMask Designer](#) (page 41).

Properties

The most common properties of the **C1InputMask** control. The **C1InputMask 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 **C1InputMask** control with a mask, enter a character to be substituted for the actual input characters
- **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.
- **Theme**
Click the drop-down arrow in the **Theme** property to select one of the built-in themes to change the appearance of the control.

- **Use CDN**

Check the **Use CDN** check box to link to the content delivery network to access Wijmo widgets and cascading style sheets.

- **CDN path**

The CDN path points to the content delivery network where you can access Wijmo widgets and cascading style sheets.

See [Using C1InputMask](#) (page 43) for details.

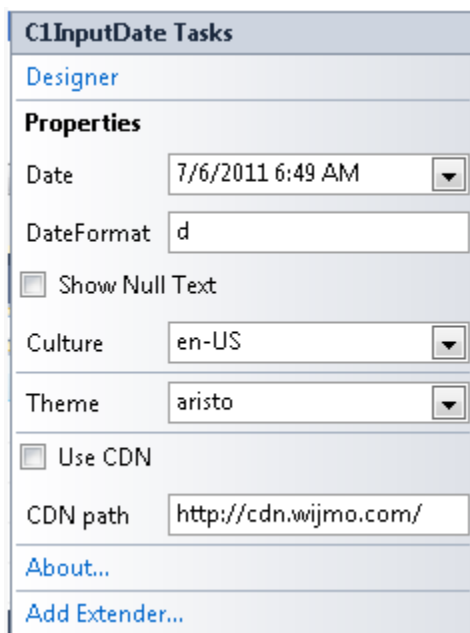
About

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

C1InputDate Smart Tag

The C1InputDate control provides quick and easy access to the most common C1InputDate properties through its smart tag.

To access the **C1InputDate Tasks** menu, click the smart tag (☰) in the upper-right corner of the C1InputDate control. This will open the **C1InputDate Tasks** menu.



The **C1InputDate Tasks** menu operates as follows:

Designer

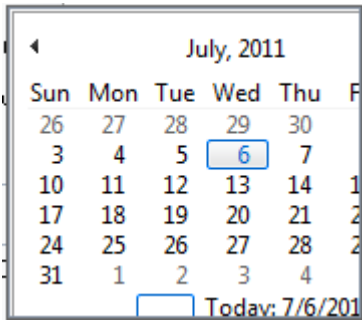
Clicking **Designer** opens the **C1InputDate Designer**. For more information on the designer, see [C1InputDate Designer](#) (page 42).

Properties

The most common properties of the **C1InputDate** control. The **C1InputDate Tasks** menu lists the following properties:

- **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*.

- **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.

- **Theme**

Click the drop-down arrow in the **Theme** property to select one of the built-in themes to change the appearance of the control.

- **Use CDN**

Check the **Use CDN** check box to link to the content delivery network to access Wijmo widgets and cascading style sheets.

- **CDN path**

The CDN path points to the content delivery network where you can access Wijmo widgets and cascading style sheets.


See [Using C1InputDate](#) (page 46) for details.

About

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

C1InputNumeric Smart Tag

The C1InputNumeric control provides quick and easy access to the most common C1InputNumeric properties through its smart tag.

To access the **C1InputNumeric Tasks** menu, click the smart tag () in the upper-right corner of the C1InputNumeric control. This will open the **C1InputNumeric Tasks** menu.

C1InputNumeric Tasks	
Properties	
Value	<input type="text" value="0"/>
MinValue	<input type="text" value="-1000000000"/>
MaxValue	<input type="text" value="1000000000"/>
DecimalPlaces	<input type="text" value="2"/>
<input type="checkbox"/> Show Null Text	
Culture	<input type="text" value="en-US"/> ▼
Theme	<input type="text" value="rocket"/> ▼
<input type="checkbox"/> Use CDN	
CDN path	<input type="text" value="http://cdn.wijmo.com/"/>
About...	

The **C1InputNumeric Tasks** menu operates as follows:

Properties

The most common properties of the **C1InputNumeric** control. The **C1InputNumeric 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.
- **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.
- **Theme**
Click the drop-down arrow in the **Theme** property to select one of the built-in themes to change the appearance of the control.
- **Use CDN**

Check the **Use CDN** check box to link to the content delivery network to access Wijmo widgets and cascading style sheets.

- **CDN path**

The CDN path points to the content delivery network where you can access Wijmo widgets and cascading style sheets.

See [Using C1InputNumeric](#) (page 49) for details.

About

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

C1InputPercent Smart Tag

The C1InputPercent control provides quick and easy access to the **most common** C1InputPercent properties through its smart tag.

To access the **C1InputPercent Tasks** menu, click the smart tag (☐) in the upper-right corner of the C1InputPercent control. This will open the **C1InputPercent Tasks** menu.

C1InputPercent Tasks	
Properties	
Value	<input type="text" value="0"/>
MinValue	<input type="text" value="-1000000000"/>
MaxValue	<input type="text" value="1000000000"/>
DecimalPlaces	<input type="text" value="2"/>
<input type="checkbox"/> Show Null Text	
Culture	<input type="text" value="en-US"/>
Theme	<input type="text" value="rocket"/>
<input type="checkbox"/> Use CDN	
CDN path	<input type="text" value="http://cdn.wijmo.com/"/>
About...	

The **C1InputPercent Tasks** menu operates as follows:

Properties

The most common properties of the **C1PerecentEdit** control. The **C1InputPercent 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.
- **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.
- **Theme**
Click the drop-down arrow in the **Theme** property to select one of the built-in themes to change the appearance of the control.
- **Use CDN**
Check the **Use CDN** check box to link to the content delivery network to access Wijmo widgets and cascading style sheets.
- **CDN path**
The **CDN path** points to the content delivery network where you can access Wijmo widgets and cascading style sheets.

See [Using C1InputPercent](#) (page 50) for details.

About

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

C1InputCurrency Smart Tag

The **C1InputCurrency** control provides quick and easy access to the common methods and properties through its smart tag.

To access the **C1InputCurrency Tasks** menu, click on the smart tag (📌) in the upper-right corner of the **C1InputCurrency** control. This will open the **C1InputCurrency Tasks** menu.

C1InputCurrency Tasks	
Properties	
Value	<input type="text" value="0"/>
MinValue	<input type="text" value="-1000000000"/>
MaxValue	<input type="text" value="1000000000"/>
DecimalPlaces	<input type="text" value="2"/>
<input type="checkbox"/> Show Null Text	
Culture	<input type="text" value="en-US"/> ▼
Theme	<input type="text" value="rocket"/> ▼
<input type="checkbox"/> Use CDN	
CDN path	<input type="text" value="http://cdn.wijmo.com/"/>
About...	

The **C1InputCurrency Tasks** menu operates as follows:

Properties

The most common properties of the **C1InputCurrency** control. The **C1InputCurrency 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.
- **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.
- **Theme**
Click the drop-down arrow in the **Theme** property to select one of the built-in themes to change the appearance of the control.
- **Use CDN**

Check the **Use CDN** check box to link to the content delivery network to access Wijmo widgets and cascading style sheets.

- **CDN path**

The CDN path points to the content delivery network where you can access Wijmo widgets and cascading style sheets.

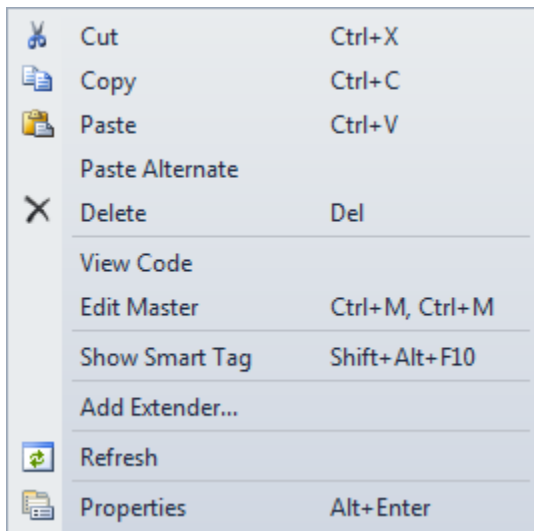
See [Using C1InputCurrency](#) (page 51) for details.

About

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

Input for ASP.NET Wijmo Context Menus

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

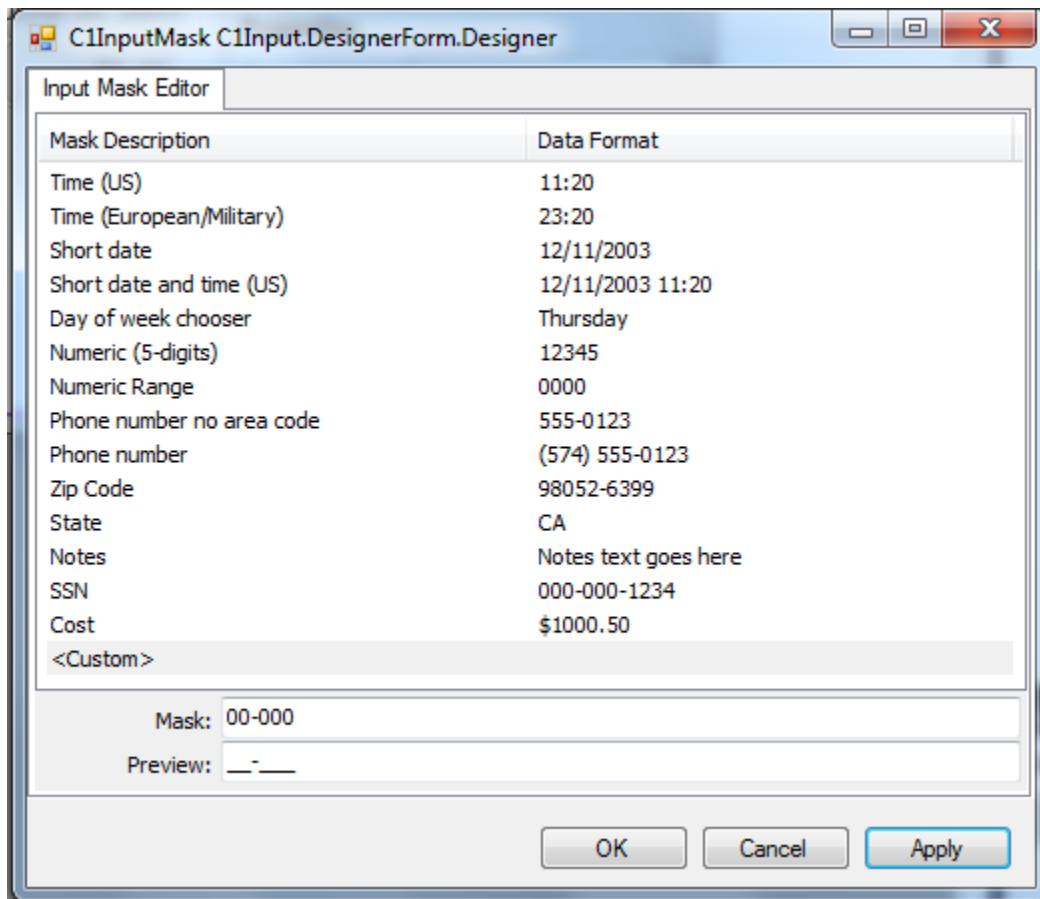


Input for ASP.NET Wijmo Designers

Input for ASP.NET Wijmo provides designers for **C1InputMask** and **C1InputDate**, allowing you to easily specify the mask or date format. These designers are described in the following topics.

C1InputMask Designer

To view the **C1InputMask Designer**, click on the smart tag (📌) in the upper-right corner of the **C1InputMask** control and select **Designer**. The following designer appears:



Input Mask Editor Tab

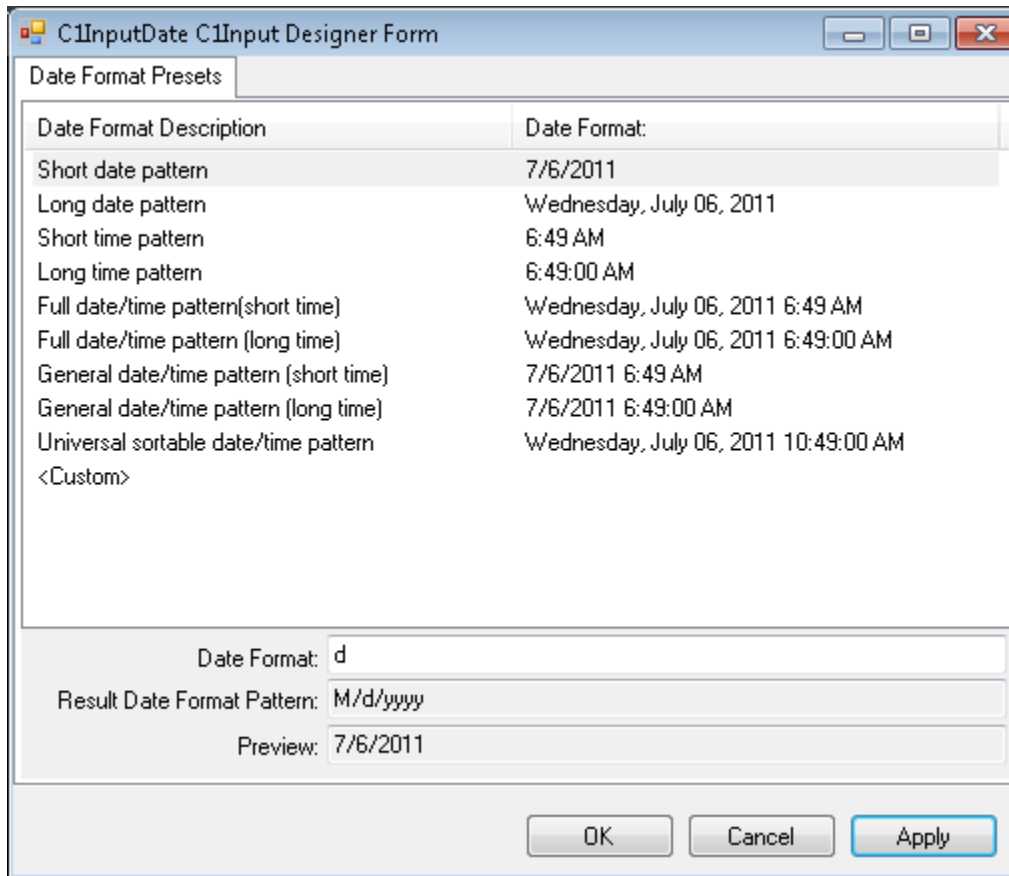
The designer's **Input Mask Editor** tab lists the C1InputMask control's mask options. The **Mask** text box shows the mask string composed of one or more placeholders (for example, 0, 9, #, and so on) and literals (for example, parentheses surrounding the area code of a telephone number). The **Preview** 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.

C1InputDate Designer

To view the **C1InputDate Designer**, click on the smart tag (E) in the upper-right corner of the **C1InputDate** control and select **Designer**.

The following designer appears:



Date Format Presets Tab

The designer's **Date Format Presets** tab lists the **C1InputDate** 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 C1InputMask

The **C1InputMask** 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. **C1InputMask** serves as a base class for the **C1InputDate** and **C1InputNumeric** controls. The following image shows a **C1InputMask** 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 **C1InputMask** 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 C1InputMask include the following:

- It's easy to master C1InputMask because its most basic properties and methods are similar in behavior with the System.Windows.Forms.MaskedTextBox control at the input of text. The C1InputMask properties and methods are distinctive with additional functionality.
- Ability to copy and paste to and from **Input for ASP.NET Wijmo** 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 **C1InputMask**, for example, English, Spanish, German, Russian, and so on.
- Ability to change most properties of **C1InputMask** "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 C1InputMask

The C1InputMask control uses a mask to distinguish between proper and improper user input. You can define the mask through the visual designers, for example, the [C1InputMask Smart Tag](#) (page 33) or the [C1InputMask Designer](#) (page 41), or programmatically through the C1InputMask object.

For common C1InputMask tasks, see the [C1InputMask Tasks](#) (page 58) topic.

C1InputMask 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 C1InputDate 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 C1InputMask 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.

C1InputMask 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:

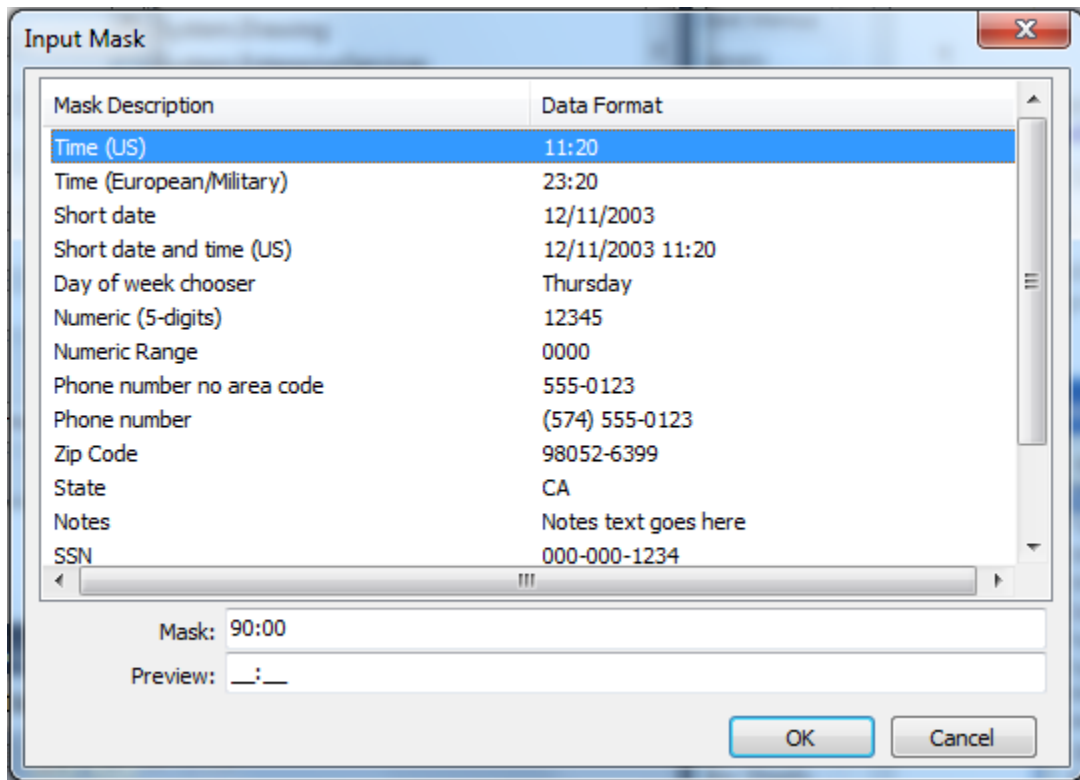
Masking Element	Description
0	Digit, required. This element will accept any single digit between 0 and 9.
9	Digit or space, optional.
#	Digit or space, optional. If this position is blank in the mask, it will be rendered as a space in the Text property. Plus (+) and minus (-) signs are allowed.
L	Letter, 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 C1InputMask . 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 C1InputMask already contains user input filtered by a previous mask, C1InputMask will attempt to migrate that input into the new mask definition.

To set the **C1InputMask.Mask** property, follow these steps:

1. Select the C1InputMask control, and click its smart tag to open the **C1InputMask Tasks** menu.
2. Click the ellipsis button next to the Mask property. The **Input Mask** dialog box appears.



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.
4. Click **OK** to close the **Input Mask** dialog box.

Using C1InputDate

The C1InputDate control, derived from C1InputMask, 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 C1InputDate control:

\$0.00

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 **C1InputDate** 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 **C1InputDate** include the following:

- **C1InputDate** control renders a date editor. Use the `DateFormat` property to set/get the date format character or pattern.
- You can set the **C1InputDate** control to interact with the **C1Calendar** control. Use the `Calendar` property to integrate **C1InputDate** with **C1Calendar**.
- Ability to choose a specific culture for **C1InputDate**, 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 C1InputDate

You can define the date pattern through the visual designers, for example, the [C1InputDate Smart Tag](#) (page 35) or the [C1InputDate Designer](#) (page 42), or programmatically through the `C1InputDate` 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.

C1InputDate General Properties

The following table lists general properties of the `C1InputDate` control:

Property	Description
<code>Date</code>	DateTime value.
<code>DateFormat</code>	Date format pattern or date format character (preset character).
<code>DateFormatResultPattern</code>	Resulting date format pattern that depends on the culture (only get).
<code>NullText</code>	Text that will be displayed for null date.
<code>ShowNullText</code>	Show null text if the <code>Date</code> value is empty and the control loses its focus.
<code>WebCalendar</code>	Gets or sets the C1WebCalendar control to interact with the C1InputDate control.

C1InputDate Format Characters

The C1InputDate 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

C1InputDate Format Patterns

The C1InputDate 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 C1InputNumeric

The **C1InputNumeric** control, derived from **C1InputMask**, 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 **C1InputNumeric** control:



Key Benefits

The key benefits of **C1InputNumeric** include the following:

- **C1InputNumeric** control renders a numeric editor.
- Ability to choose a specific culture for **C1InputNumeric**, for example, English, Spanish, German, Russian, and so on. Note that **C1InputNumeric** 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 C1InputNumeric

The **C1InputNumeric** control has numeric range support for displaying numerical data. Since the **C1InputNumeric** control deals strictly with numbers, there is no input mask to specify. To define the **C1InputNumeric**, 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 C1InputNumeric control through the [C1InputNumeric Smart Tag](#) (page 36) or programmatically through the C1InputNumeric 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 C1InputNumeric tasks, see the [C1InputNumeric Tasks](#) (page 67) topic.

C1InputNumeric General Properties

The following table lists general properties of the C1InputNumeric control:

Property	Description
Value	Double, numeric value of the C1InputNumeric 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 C1InputNumeric control are the same as the C1InputMask control, except for hidden properties that are not used in numeric controls (such as the following: AllowPromptAsInput, Mask, HidePromptOnLeave, PasswordChar, PromptChar, ResetOnPrompt, ResetOnSpace, SkipLiterals).

Using C1InputPercent

The C1InputPercent control, derived from C1InputNumeric, 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 C1InputPercent control:



Key Benefits

The key benefits of C1InputPercent include the following:

- **C1InputPercent** control renders a numeric editor. C1InputPercent can be used to input percent values.

- Ability to choose a specific culture for C1InputPercent, 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 C1InputPercent control's properties are the same as the C1InputNumeric control.

Defining C1InputPercent

The C1InputPercent control has numeric range support for displaying numerical data. You can define the value of the C1InputPercent control through the [C1InputPercent Smart Tag](#) (page 38) or programmatically through the C1InputPercent 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 C1InputPercent control's properties are the same as the C1InputNumeric control. For common C1InputPercent tasks, see the [C1InputNumeric Tasks](#) (page 67) topic.

Using C1InputCurrency

The C1InputCurrency control, derived from C1InputNumeric, 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 C1InputCurrency control:



Key Benefits

The key benefits of C1InputCurrency include the following:

- C1InputCurrency control renders a numeric editor. C1InputCurrency can be used to input currency values.
- Ability to choose a specific culture for C1InputCurrency, 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 C1InputCurrency control's properties are the same as the C1InputNumeric control.

Defining C1InputCurrency

The C1InputCurrency control has numeric range support for displaying numerical data. You can define the value of the C1InputCurrency control through the [C1InputCurrency Smart Tag](#) (page 39) or programmatically through the C1InputCurrency 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 C1InputCurrency control's properties are the same as the C1InputNumeric control. For common C1InputCurrency tasks, see the [C1InputNumeric Tasks](#) (page 67) topic.

Input for ASP.NET Wijmo Appearance

Change the input control's look by selecting one of the six premium themes (*Arctic*, *Midnight*, *Aristo*, *Rocket*, *Cobalt*, and *Sterling*). Or you can use ThemeRoller from jQuery UI to create your own customized theme!

Built-in Wijmo Themes

Input for ASP.NET Wijmo provides six built-in themes for each **Input for ASP.NET Wijmo** control, allowing you to automatically format the controls. The themes include the following: **arctic**, **aristo**, **cobalt**, **midnight**, **rocket**, and **sterling**.

The examples below show the C1InputMask control; however, the themes for all controls are the same.

arctic

The following image displays the **arctic** theme:



aristo

The following image displays the **aristo** theme. This is the default format for all of the **Input for ASP.NET Wijmo** controls:



cobalt

The following image displays the **cobalt** theme:



midnight

The following image displays the **midnight** theme:



rocket

The following image displays the **rocket** theme:



sterling

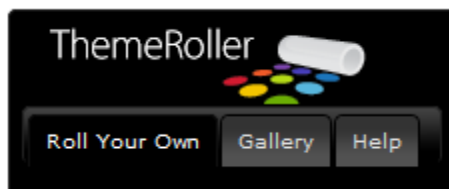
The following image displays the **sterling** theme:

() -

ThemeRoller Overview

One of the most convenient features of Wijmo is its ThemeRoller compatibility. ThemeRoller is a Web application for creating unique jQueryUI themes to skin your Web app's widgets. With its simple interface and WYSIWYG preview, you can create a skin for all of your Wijmo widgets and other ThemeRoller-compatible widgets in less time than it would take to open a graphic editor.

We can find the ThemeRoller Web app at <http://jqueryui.com/themeroller/>. Once you've arrived at the app, take a gander at the column on the left-hand side of the page and observe that it holds three tabs: **Roll Your Own**, **Gallery**, and **Help**.



The Roll Your Own Tab

Roll Your Own is where the magic happens. From this tab, we have the power to tweak your theme to perfection. To change an element, all we have to do is expand a node and get to work.

The Gallery Tab

Clicking the **Gallery** contains a list and preview of ThemeRoller's premium themes. From here, we can preview, download, or edit one of the ready-made themes in the following ways:

- Clicking the snippet view of the theme loads an interactive preview to the right of the gallery.
- Clicking a theme's **Download** button will take us to the **Build Your Download** page. If we want to download one of the premade themes to skin your Wijmo widgets, all we have to do is navigate to the light orange panel, select our **Advanced Theme Settings** and **Version**, and click **Download**.
- Clicking **Edit** will load the chosen theme and return us to the **Roll Your Own** tab, where we can use the ThemeRoller to tweak the theme.

The Help Tab

Clicking the **Help** tab will give us a quick reference and special information regarding the ThemeRoller, such as information for plugin developers and browser support notices.

See [Adding a Custom Theme](#) (page 69) for instructions on using the ThemeRoller to download a theme.

For more information on using the ThemeRoller and for a short tutorial, visit our [Wijmo website](#).

C1Input CSS Selectors

You can style any **C1Input** elements using CSS styles to make their appearance truly unique. To make the customization process easier, ComponentOne includes CSS selectors for each of its six built-in themes. For more information on themes, see [Built-in Wijmo Themes](#) (page 52).

You can apply general CSS properties such as background, text, font, border, outline, margin, padding, list, and table to applicable CSS selectors.

The following topic details the common individual CSS selectors and grouped CSS selectors. You can combine the individual CSS selectors as a group to make the CSS selector more specific and strong.

CSS Selectors	Description
.wijmo-input-trigger	Applies the style to the trigger button.
.wijmo-wijinput-spinner	Applies the style to the spinner button.
.wijmo-wijinput-spinup	Applies the style to the up spin button.
.wijmo-wijinput-spindown	Applies the style to the down spin button.
.wijmo-wijinput-input	Applies the style to the outmost container for all the input types.
.wijmo-wijinput-mask	Applies the style to the outmost container for the mask input (C1InputMask).
.wijmo-wijinput-numeric	Applies the style to the outmost container for the number input types (C1InputNumeric, C1InputCurrency and C1InputPercent).
.wijmo-wijinput-date	Applies the style to the outmost container for the date input types (C1InputDate).
.wijmo-wijinput-wrapper	Applies the style to the direct wrapper for the input element.
.wijmo-wijinput ui-state-focus	Applies the style to the outmost container for all the input types when it is in focused state.

Working with the Client-Side

The **Input for ASP.NET Wijmo** controls have a very rich client-side object model since most of their members are identical to the members in the server-side control.

When a **C1Input** control is rendered, an instance of the client-side control will be created automatically. This means that you can enjoy the convenience of accessing properties and methods of the **C1Input** controls without having to postback to the server.

Using client-side code, you can implement many features in your Web page without the need to send information to the Web server, which takes time. Thus, using the client-side object model can increase the efficiency of your Web site.

Client-Side Events

Input for ASP.NET Wijmo includes several client-side events that allow you to manipulate the **C1Input** controls when an action such as an invalid character is entered.

You can use the sever-side properties, listed in the Client Side Event table, to specify the name of the JavaScript function that will respond to a particular client-side event. For example, to assign a JavaScript function called **invalidInput** to respond when an invalid character is entered, you would set the OnClientInvalidInput property to **invalidInput**.

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

Event Server-Side Property Name	Event Name	Description
OnClientInitialized	initialized	Occurs after the widget is initialized.
OnClientInitializing	initializing	Occurs before the widget is initialized.
OnClientInvalidInput	invalidInput	Occurs when an invalid character is entered.
OnClientTextChanged	textChanged	Occurs when the text of the input is changed.
OnClientTriggerMouseDown	triggerMouseDown	Occurs when the mouse is pressed down on the trigger button.
OnClientTriggerMouseUp	triggerMouseUp	Occurs when the mouse is released on the trigger button.
OnClientDateChanged (InputDate only)	dateChanged	Occurs after the date value changed.
OnClientValueBoundsExceeded (InputNumber only)	valueBoundsExceeded	Occurs when the value of the input exceeds the valid range.
OnClientValueChanged (InputNumber only)	valueChanged	Occurs after the value changed.

For task-based help on client-side events, see the [Client-Side Event Tasks](#) (page 72) topic. Descriptions and syntax examples for the **C1Input** client-side events can also be found at <http://wijmo.com/wiki/index.php/InputMask>.

Input for ASP.NET Wijmo 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 Wijmo | Control Explorer**.

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

C# Sample

Sample	Description
C1InputDate:	
DatePicker	This sample demonstrates that when the showTrigger property is set to true, clicking the trigger button opens the default calendar to be used as a date picker.
DropDown	This sample shows how you can easily create a drop-down box for date input.
Overview	This sample shows the C1InputDate control, an input control that is specialized for editing Date/Time values.
C1InputMask:	
DropDown	This sample shows how predefined input values can be specified using the ComboItems property.
FirstName	This sample illustrates how you can use the Mask property to allow only text input.
Overview	This samples shows the C1InputMask , an input web control that allows users to type text based on the mask.
Password	This sample demonstrates how to display password characters, such as * and #, by setting the PasswordChar property.
C1InputNumber:	
Currency	The C1InputCurrency control is an input control that is specialized for editing currency values.
DropDown	This sample demonstrates how you can create a drop-down list with currency items by setting the ComboItems properties.
Increment	This sample demonstrates how to use the Increment property to create a custom increment.
Overview	The C1InputNumeric control is an input control that is specialized for editing numeric values.
Percent	The C1InputNumeric control is an input control that is specialized for editing numeric values.

Input for ASP.NET Wijmo 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 Wijmo features and get a good sense of what Input for ASP.NET Wijmo can do.

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

C1InputMask Tasks

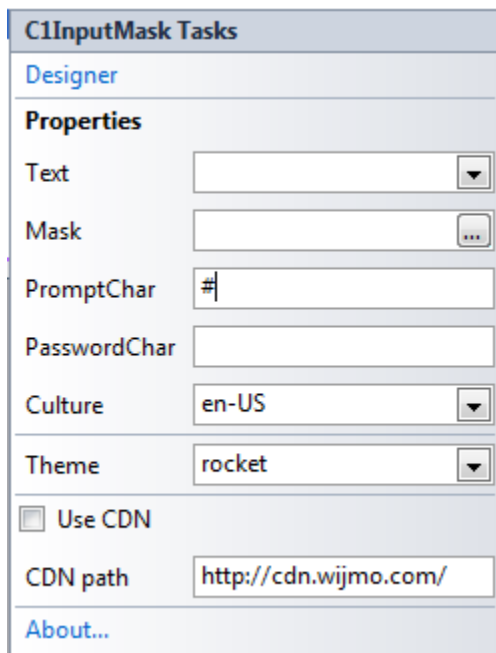
This section shows how to perform specific tasks using the C1InputMask control. The following topics assume that you have added a C1InputMask control to your Web form.

Changing the Prompt Character

At run time, the C1InputMask 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 **C1InputMask** 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 **C1InputMask Tasks** menu and enter the number sign (#) in the **PromptChar** text box.



To change the prompt character using .html markup:

To change the prompt character to the number sign (#) for the **C1InputMask** control, use the following markup in the .aspx page:

```
<wijmo:C1InputMask ID="C1InputMask1" runat="server"
```

```
Mask="(999) 000-0000"  
Text="412"  
PromptChar="#">  
</wijmo:C1InputMask>
```

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:



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

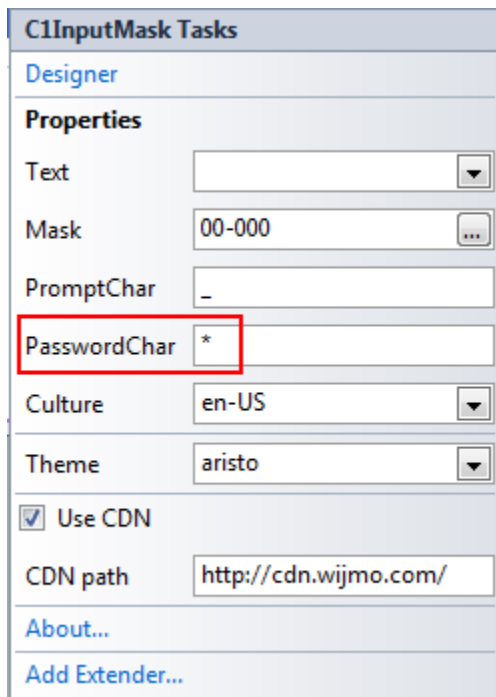
For details on hiding the prompt characters when the input box loses focus, see the [Hiding the Prompt Character on Leave](#) (page 63) topic.

Using Password Protection

You can protect input text by displaying password characters for the C1InputMask control so that the actual characters entered are not visible.

To set password characters using the Tasks menu:

Open the **C1InputMask Tasks** menu and enter an asterisk (*) in the **PasswordChar** text box.



To change the password character using .html markup:

To change the password character to an asterisk (*) for the **C1InputMask** control, use the following markup in the .aspx page:

```
<wijmo:C1InputMask ID="C1InputMask1" runat="server" Mask="00-000" Width="200px"
```

```
        PasswordChar="*">
    </wijmo:C1InputMask>
```

This topic illustrates the following:

Run the project and enter characters in the C1InputMask control. Notice that an asterisk (*) is displayed for each character as it is entered, as shown here:

Product Number:

Creating Day of the Week Chooser Mask

The following example demonstrates using enumeration parts in the Mask property. This example uses the C1InputMask 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 **C1InputMask Tasks** menu and click **Designer** to open the **C1InputMask 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.

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:

```
<wijmo:C1InputMask ID="C1InputMask1" runat="server"
Mask="&lt;&lt;Monday|Tuesday|Wednesday|Thursday|Friday&gt;&gt;"
Text="Wednesday"
/>
```

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 C1InputMask control with the custom mask:

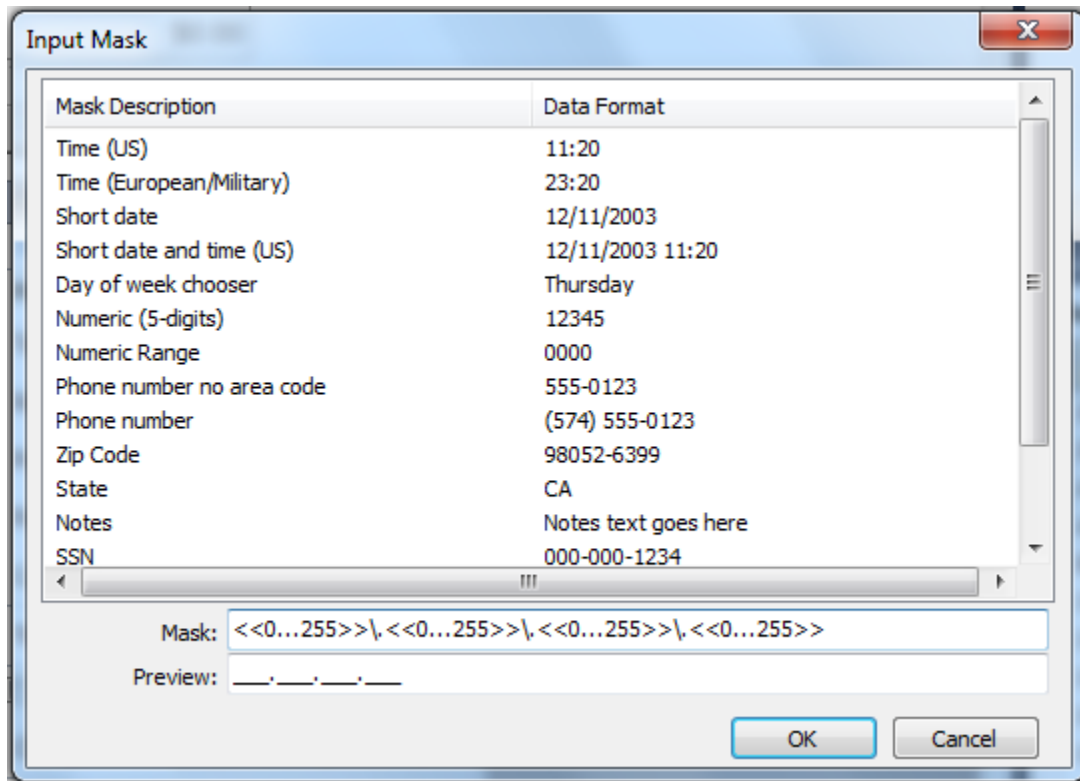
```
<<0...255>>\.<<0...255>>\.<<0...255>>\.<<0...255>>.
```

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 **C1InputMask 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>>



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:

```
<wijmo:C1InputMask ID="C1InputMask1" runat="server"
Mask="&lt;&lt;0...255&gt;&gt;\.&lt;&lt;0...255&gt;&gt;\.&lt;&lt;0...255&gt;&gt;\.&lt;&lt;0...255&gt;&gt;"
Text="192168001001">
</wijmo:C1InputMask>
```

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 C1InputMask 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 **C1InputMask 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.

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:

```
<wijmo:C1InputMask ID="C1InputMask1" runat="server"
    Mask="(999) 000-0000"
    Text="412"
</wijmo:C1InputMask>
```

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 C1InputMask 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 **C1InputMask 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:

```
<wijmo:C1InputMask ID="C1InputMask1" runat="server" Mask="00/00/0000"
PromptChar=" ">
</wijmo:C1InputMask>
```

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

To create a short date mask without prompt characters for the **C1InputMask** 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 C1InputMask1
    .Mask = "00/00/0000"
    .PromptChar = " "
End With
```

- C#

```
this.C1InputMask1.Mask = "00/00/0000";
this.C1InputMask1.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:



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:

```
<wijmo:C1InputMask ID="C1InputMask1" runat="server"
Mask="(999) 000-0000"
PromptChar="#"
HidePromptOnLeave="True">
</wijmo:C1InputMask>
```

To hide the prompt character on leave using code:

To hide the prompt character on leave for the **C1InputMask** 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 C1InputMask1
    .Mask = "(999) 000-0000"
    .PromptChar = "#"
    .HidePromptOnLeave = True
End With
```

- C#

```
this.C1InputMask1.Mask = "(999) 000-0000";
this.C1InputMask1.PromptChar = char.Parse("#");
this.C1InputMask1.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 58) topic.

C1InputDateTasks

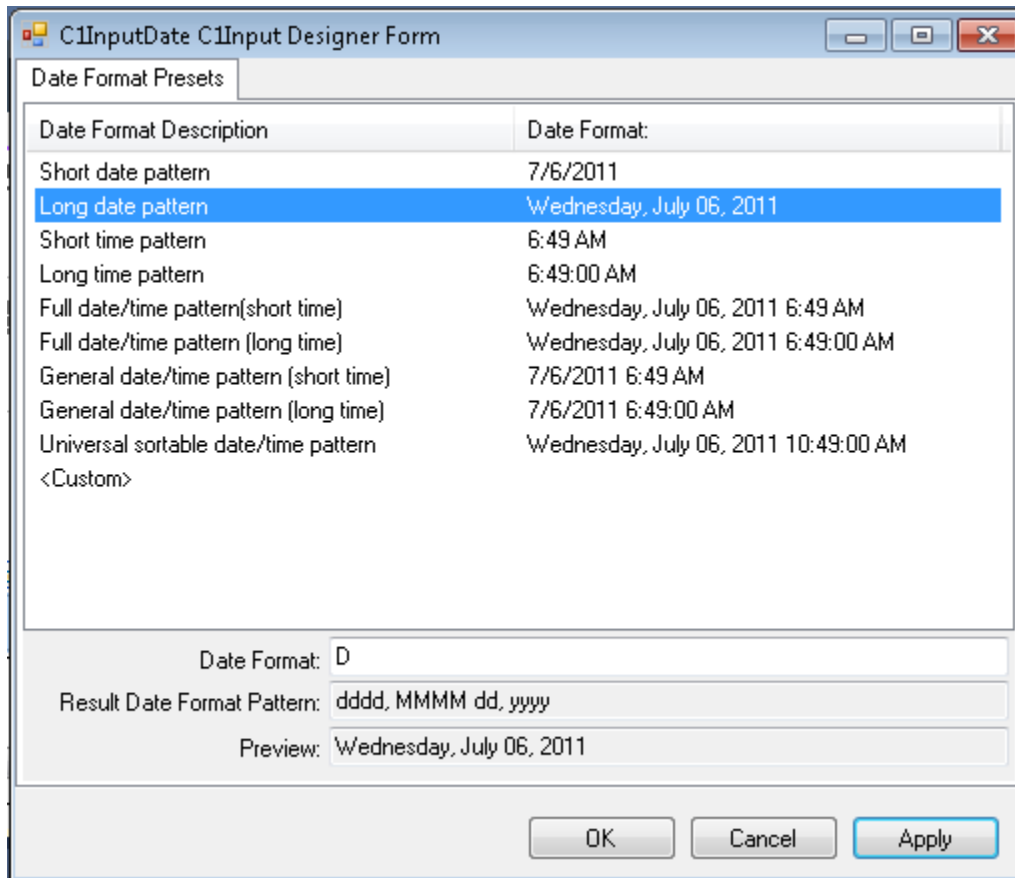
This section shows how to perform specific tasks using the C1InputDate control. The following topics assume that you have added a C1InputDate control to your Web form.

Setting the Date Format Pattern and Date

The following example demonstrates how to set the date format pattern for the **C1InputDate** control.

To set the date format pattern using the Tasks menu:

1. Open the **C1InputDate Tasks** menu and click **Designer**. The **C1InputDate Designer** appears.
2. Choose a preformatted date pattern. For this example, select **Long date pattern**.



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:

```
<wijmo:C1InputDate ID="C1InputDate1" runat="server"
  Date="2006-12-19"
  DateFormat="D">
</wijmo:C1InputDate>
```

To set the date format pattern using code:

To set the date format pattern for the C1InputDate control, double-click the Web page to create an event handler for the **Load** event. Enter the following code for the **Page_Load** event:

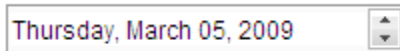
- Visual Basic


```
' Format the control as long date pattern
Me.C1InputDate1.DateFormat = "D"
' Set the date
Me.C1InputDate1.Date = "2006-12-19"
```
- C#

```
// Format the control as long date pattern
this.C1InputDate1.DateFormat = "D";
// Set the date
this.C1InputDate1.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

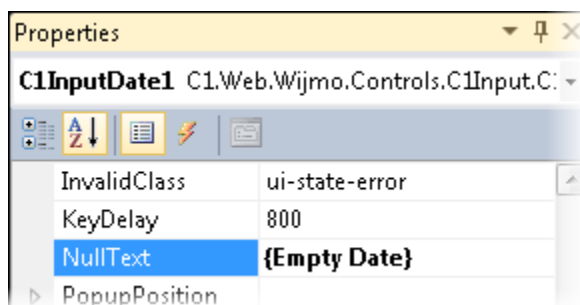
When you run your project that contains a C1InputDate control, the default date, January 1, 0001 12:00:00, automatically appears inside the control, regardless of how you have customized the controls (see [Setting the Date Format Pattern and Date](#) (page 64)). If you want the date to remain empty and not to show this default January 1, 0001 date, complete the following steps:

Using the Designer

1. Place a C1InputDate 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 C1InputDate control to another control on your page.



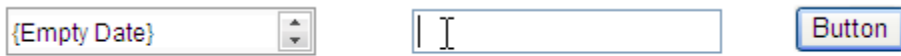
2. Switch the ShowNullText property from **False** to **True**.
3. Enter the desired Empty Date Value in the NullText property. For this example, use **{Empty Date}**.



4. Run the project and notice the **{Empty Date}** of the C1InputDate control.
5. Select the C1InputDate 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 C1InputDate 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.

C1InputNumeric Tasks

This section shows how to perform specific tasks using the C1InputNumeric control.

Note that the C1InputCurrency and C1InputPercent control's properties are the same as the C1InputNumeric control; therefore, the following tasks apply to the C1InputCurrency and C1InputPercent controls as well.

The following topics assume that you have added a C1InputNumeric control to your Web form.

Indicating the Number of Decimal Places

The following example shows how you can easily indicate the number of decimal places to display for the C1InputNumeric control.

To set the decimal places value using the Tasks menu:

1. Open the **C1InputNumeric 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:

```
<wijmo:C1InputNumeric ID="C1InputNumeric1" runat="server"
    DecimalPlaces="3"
    Value="2.345">
</wijmo:C1InputNumeric>
```

To set the decimal places value using code:

To set the decimal places value for the C1InputNumeric 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.C1InputNumeric1.Value = 2.345
' Set the number of decimal places
Me.C1InputNumeric1.DecimalPlaces = 3
```
- C#

```
// Set the numeric value
this.C1InputNumeric1.Value = 2.345;
// Set the number of decimal places
```

```
this.C1InputNumeric1.DecimalPlaces = 3;
```

Setting the Min/Max Value

The following example demonstrates the C1InputNumeric 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 **C1InputNumeric 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:

```
<wijmo:C1InputNumeric ID="C1InputNumeric1" runat="server"
    MaxValue="1000"
    MinValue="1"
    Value="1">
</wijmo:C1InputNumeric>
```

To set the numeric value using code:

To set the numeric value for the **C1InputNumeric** 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 C1InputNumeric1
    .MaxValue = 1000
    .MinValue = 1
    .Value = 1
End With
```

- C#

```
this.C1InputNumeric1.MaxValue = 1000;
this.C1InputNumeric1.MinValue = 1;
this.C1InputNumeric1.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.

Changing the Theme

You can format the **Input for ASP.NET Wijmo** controls with one of six built-in themes. We will use C1InputMask in the following examples.

Changing the Theme Using the Smart Tag

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

1. Click the C1InputMask smart tag to open the **C1InputMask Tasks** menu.

2. Click drop-down arrow next to **Theme**.
3. Select one of the built-in themes listed. The theme is applied to the C1InputMask control.

Changing the Visual Style in Code

To change the style of an **Input for ASP.NET Wijmo** control programmatically, use the following code. In this example, **midnight** is used, but you can replace it with any of the built-in themes.

- Visual Basic

```
C1InputMask1.VisualStudio = "midnight"
```
- C#

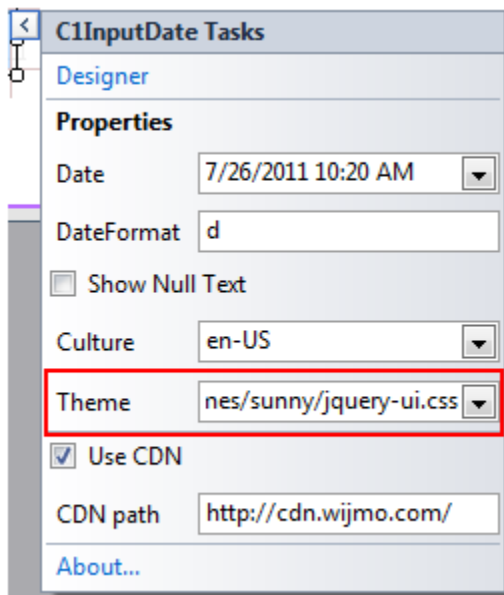
```
C1InputMask1.VisualStudio = "midnight";
```

Adding a Custom Theme

Input for ASP.NET Wijmo provides six built-in themes, but if you prefer to use a different theme, you can choose an existing theme using a CDN URL or create your own custom theme with the jQuery ThemeRoller Web application. We will use C1InputDate in the following examples.

Using a CDN URL

1. Click the C1InputDate smart tag to open the **Tasks** menu.
2. Select **Use CDN**.
3. In the **Theme** property, enter a CDN URL to specify the theme; CDN URLs can be found at <http://blog.jqueryui.com/2011/06/jquery-ui-1-8-14/>. In this example, we'll use the *sunny* theme: <http://ajax.aspnetcdn.com/ajax/jquery.ui/1.8.14/themes/sunny/jquery-ui.css>.



This theme setting is stored in the **<appSettings>** of the **Web.config** file. In the Solution Explorer, double-click the **Web.config** file. Notice the **<appSettings>** tag contains a **WijmoTheme** key and value; this is where the CDN URL you added is specified.

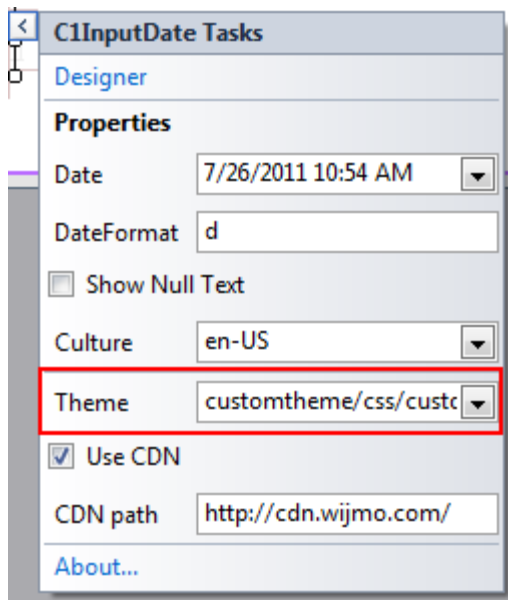
4. Run the project and notice the theme is applied to C1InputDate.

Sunny Theme

7/26/2011

Using jQuery ThemeRoller

1. Go to <http://jqueryui.com/themeroller/>.
2. On the **Roll Your Own** tab, change the settings to create a custom theme; you can customize fonts, colors, backgrounds, borders, and more. Or click the **Gallery** tab and select an existing theme.
3. Click the **Download** button and then click **Download** again on the **Build Your Download** page.
4. Save and unzip the theme .zip file to a folder within your Visual Studio project folder. In this example, we created a **customtheme** folder.
5. In the Solution Explorer, click **Show All Files** and then right-click the **customtheme** folder and select **Include in Project**.
6. Click the C1InputDate smart tag to open the **Tasks** menu.
7. Select **Use CDN**.
8. In the **Theme** property, enter the path to your custom theme .css, for example, **customtheme/css/custom-theme/jquery-ui-1.8.14.custom.css**.



This theme setting is stored in the **<appSettings>** of the **Web.config** file. In the Solution Explorer, double-click the **Web.config** file. Notice the **<appSettings>** tag contains a **WijmoTheme** key and value; this is where the custom theme you added is specified.

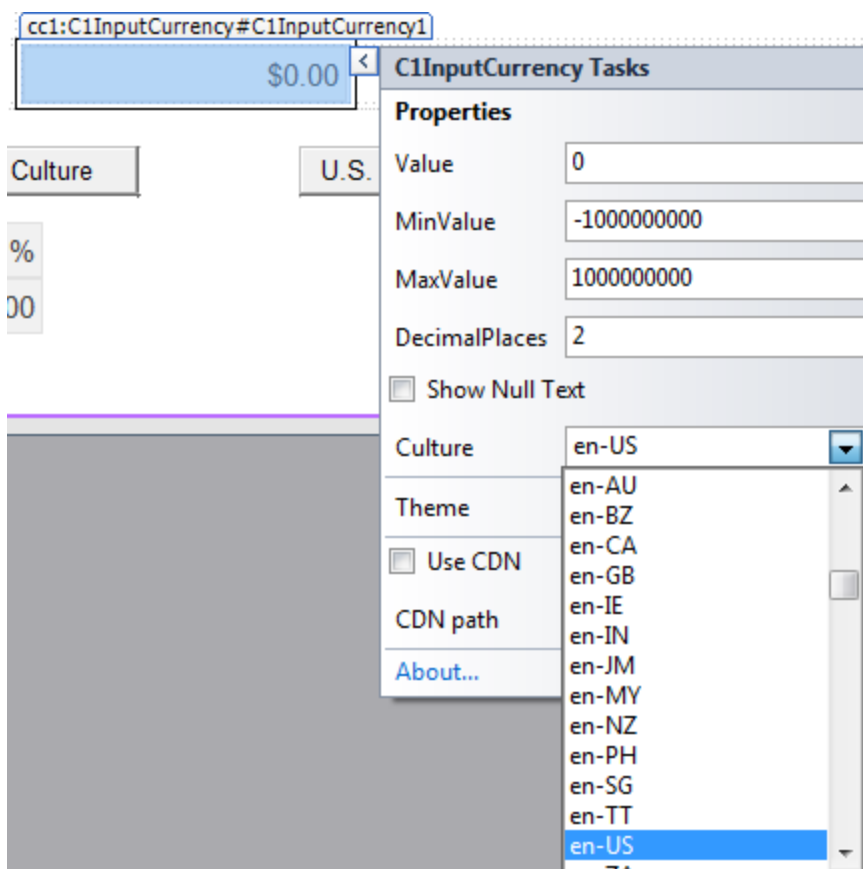
9. Run the project and notice the theme is applied to C1InputDate.

Selecting the Culture

Note that the following topic shows how to use the Culture property for the **C1InputCurrency** control; however, the Culture property is available for all **Input for ASP.NET Wijmo** controls.

Using the Designer

You can choose a specific culture for any of the **Input for ASP.NET Wijmo** 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:

```
<wijmo:C1InputCurrency ID="C1InputCurrency1" runat="server"
    Culture="English (United Kingdom)">
</wijmo:C1InputCurrency>
```

Using Code

To set the Culture for the **C1InputCurrency** 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.C1InputCurrency1.Culture = New System.Globalization.CultureInfo("en-GB")
```

- C#

```
this.C1InputCurrency1.Culture = new System.Globalization.CultureInfo("en-GB");
```

This topic illustrates the following:

The following **C1InputCurrency** control shows the British Pound:



Client-Side Event Tasks

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

Showing a Tooltip when Invalid Input is Entered

This topic demonstrates how to show a tooltip, we'll use a **wijtooltip** in this example, when an invalid character is entered into the **C1InputMask** control.

1. Right-click the **C1InputMask** control on your form and choose **Properties** to open the Visual Studio Properties window.
2. Next to the **OnClientInvalidInput** property, enter **invalidInput**.
3. Select the **Source** tab to open the source view.
4. In the .aspx source, enter the following script markup:

```
<script type="text/javascript">
    function invalidInput(e, data) {
        $(data.widget.element).wijtooltip({
            title: '\"' + data.char + '\" is invalid to mask ' +
data.widget.options.mask,
            triggers: 'custom',
            showing: function () {
                window.setTimeout(function () {
                    $(data.widget.element).wijtooltip('hide');
                }, 3000);
            }
        });
        $(data.widget.element).wijtooltip('show');
    }
</script>
```

When an invalid character is entered in the **C1InputMask** control, a tooltip appears, like in the following image:

Using a Trigger to Show a Custom UI

This topic demonstrates how to show a custom interface when a trigger button is clicked. In this example, the user will click the arrow in a **C1InputNumeric** control, and a drop-down slider will appear and change the input value when the slider thumb is dragged.

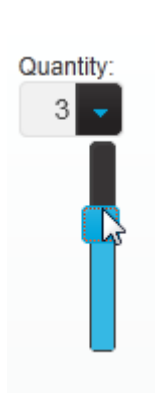
1. Right-click the **C1InputNumeric** control on your form and choose **Properties** to open the Visual Studio Properties window.
2. Next to the OnClientTriggerMouseDown property, enter **triggerClicked**.
3. Select the **Source** tab to open the source view.
4. In the .aspx source, enter the following script markup:

```
function triggerClicked(e) {
    var $input = $('#<%=C1InputNumeric1.ClientID%>');
    var val = $input.clinputnumeric('option', 'value');

    var $volumeSlider = $('.valueslider');
    $volumeSlider.slider({
        min: 0,
        max: 5,
        value: val,
        step: 1,
        orientation: 'vertical',
        range: 'min',
        slide: function (e, ui) {
            $input.clinputnumeric('option', 'value',
ui.value);
        }
    });

    $(".dropdown-container").wijpopup('show', {
        of: $('.wijmo-wijinput'),
        at: 'right bottom',
        my: 'right top',
        offset: "4 2"
    });
}
```

When a user clicks the drop-down arrow, a slider pops up. When the slider is moved, the quantity changes, like in the following image:



Input for ASP.NET Wijmo Client-Side Reference

As part of the amazing [ComponentOne Web stack](#), the Wijmo jQuery UI widgets are optimized for client-side Web development and utilize the power of jQuery for superior performance and ease of use.

The ComponentOne Wijmo website at <http://wijmo.com/widgets/> provides everything you need to know about Wijmo widgets, including demos and samples, documentation, theming examples, support and more.

The client-side documentation provides an overview, sample markup, options, events, and methods for each widget. To get started with client-side Web development for **Input for ASP.NET Wijmo**, click one of the external links to view our Wijmo wiki documentation. Note that the **Overview** topic for each of the widgets applies mainly to the widget, not to the server-side ASP.NET Wijmo control.

- InputMask
 - [wijinputmask dependencies](#)
 - [wijinputmask options](#)
 - [wijinputmask events](#)
- InputNumber
 - [wijinputnumber dependencies](#)
 - [wijinputnumber options](#)
 - [wijinputnumber events](#)
- InputDate
 - [wijinputdate dependencies](#)
 - [wijinputdate options](#)
 - [wijinputdate events](#)

Using the Wijmo CDN

You can easily load the client-side Wijmo widgets into your web page using a Content Delivery Network (CDN). CDN makes it quick and easy to use external libraries, and deploy them to your users. A CDN is a network of computers around the world that host content. Ideally, if you're in the United States and you access a webpage using a CDN, you'll get your content from a server based in the US. If you're in India or China, and you access the SAME webpage, the content will come from a server a little closer to your location.

When web browsers load content, they commonly will check to see if they already have a copy of the file cached. By using a CDN, you can benefit from this. If a user had previously visited a site using the same CDN, they will already have a cached version of the files on their machine. Your page will load quicker since it doesn't need to re-download your support content.

Wijmo has had CDN support from the very beginning. You can find the CDN page at <http://wijmo.com/downloads/cdn/>. The markup required for loading Wijmo into your page looks similar to this:

```
<!--jQuery References-->
<script src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-1.7.1.min.js"
type="text/javascript"></script>
<script src="http://ajax.aspnetcdn.com/ajax/jquery.ui/1.8.17/jquery-
ui.min.js" type="text/javascript"></script>
<!--Theme-->
```

```
<link href="http://cdn.wijmo.com/themes/rocket/jquery-wijmo.css"
rel="stylesheet" type="text/css" title="rocket-jqueryui" />
<!--Wijmo Widgets CSS-->
<link href="http://cdn.wijmo.com/jquery.wijmo-complete.all.2.0.0.min.css"
rel="stylesheet" type="text/css" />
<!--Wijmo Widgets JavaScript-->
<script src="http://cdn.wijmo.com/jquery.wijmo-open.all.2.0.0.min.js"
type="text/javascript"></script>
<script src="http://cdn.wijmo.com/jquery.wijmo-complete.all.2.0.0.min.js"
type="text/javascript"></script>
```

In this markup, you'll notice that some of the .js files are labeled as *.min.js. These files have been minified - in other words, all unnecessary characters have been removed - to make the pages load faster. You will also notice that there are no references to individual .js files. The JavaScript for all widgets, CSS, and jQuery references have been combined into one file, respectively, such as wijmo-complete.2.0.0.min.js. If you want to link to individual .js files, see the **Dependencies** topic for each widget.

With the **ComponentOne Studio for ASP.NET Wijmo** controls, you can click the **Use CDN** checkbox in the control's **Tasks** menu and specify the **CDN path** if you want to access the client-side widgets.

